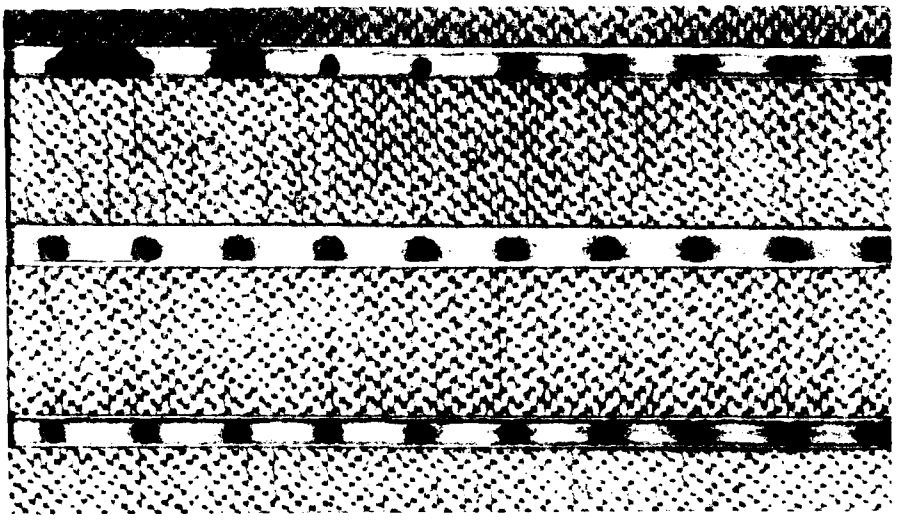
(12)



N00014-84-K-0482
MAY 1986

Department of

ELECTRICAL ENGINEERING

UNIVERSITY OF NOTRE DAME, NOTRE DAME, INDIANA

(12)

Analytical Models for Parallel Processing Systems

H. Ammar, Y. F. Huang and R. Liu

Department of Electrical & Computer Engineering
University of Notre Dame
Notre Dame, IN 46556

EE Technical Report-#861

EXECUTIVE SUMMARY

Evaluations of reliability, maintainability, and availability (RMA) of large-scale complex systems have received a great deal of attention in defense and commercial fields. In these studies, an extremely difficult yet critical issue is effectiveness of the model.

Experience in RMA anlaysis for many practical large-scale systems has shown that more than 50% of BIT-related maitenance actions are due to false alarms. This clearly implies an excessive operation and support (O&S) costs. Further, to improve system availability, one often employs redundant components (or modules). Redundancy not only increases hardware costs but imposes additional difficulties on analyzing system RMA as it increases modeling complexity, especially for large-scale systems.

Analytical models for such systems that provide an accurate picture yet are not too complicated are very difficult to find. Simulations, though can be made very accurate, could often be costly. On the other hand, analytical models are very efficient for sensitivity analysis and numerous tradeoff studies, provided that they are accurate.

Two important ingredients must be taken into account in setting up models for RMA analysis, i.e., conditions and activities (events) of the system. In fact, for large-scale systems, numbers of conditions and activities often become intractably large. It is this problem that has prevented most currently available schemes from providing accurate and effective RMA analysis.

In this report, we propose to use generalized stochastic Petri nets (GSPN) in RMA studies. The novelty of this modeling approach lies on the ground of the following distinctive reasons.

(i) The GSPN offers a precise description of system activities and conditions while involves less complexity, comparing to other modeling techniques. Specifically, it is an inherently effective bookkeeping for conditions and activities.

(ii) It provides a clairvoyant insight of the key parameters that affect RMA analysis. Causes and results of events can be easily tracked by executing the GSPN.

(iii) It takes the advantage of the existence of concurrency and timing of events, thus describes accurately the sequence of events.

The report is divided into two parts. In the first part, definitions and classifications of concurrent tasks are given. Existing analytical models which are based on queueing networks (QN) are reviewed. Approximate hierarchical models based on GSPN and QN are both presented. In the second part, analysis of GSPN is considered. Techniques for reducing analytical complexity such as reduction and aggregation of GSPN are introduced. Applicaitons of such techniques to the approximate hierarchical decomposition of stochastic Petri nets are discussed. In addition, approximate lumping of synchronous parallel operations is considered. For simplicity of discussions, many of these results are illustrated via performance evaluation of computer systems. Finally, examples of RMA analysis and fault detection and isolation are given using the developed techniques.

TABLE OF CONTENTS

PART II: ANALYSIS

ACKNOWLEDGEMENT

This research has been supported in part by the Naval Air Systems Command under an ONR Contract N00014-84-K-0482. The authors wish to express their appreciation to James G. Smith of Naval Air Systems Command and Clifford Lau of Office of Naval Research for their assistance in the course of this work. Special thanks are due to Randall L. Fleming for valuable discussions.

CHAPTER 1

INTRODUCTION

1.1 Motivation

Advances in solid-state technology have provided us with high speed computer systems of ever increasing computational power. In addition to the speed of the components, their organization may be a limiting factor. Design efforts have therefore been geared toward improving performance on the system level. Parallelism in the architecture has been the most successful approach. It includes multiple functional units, pipelining array structures, and multiprocessor architectures. Distributed computing and network configuration represent parallelism on an even higher level. Design efforts also focus on operating system functions and strategies for managing system resources, and further improvements can yet be obtained by designing programming languages features that match the underlying architecture. Common to all these design efforts is the desire to evaluate perfomance impacts prior to implementation. Even though the basic components of such systems are inexpensive, the design costs are so high that an incorrect design which is undetected until late in the development process, can have a serious negative impact on a company. Therefore, cost effective tools for performance prediction of such system , at the early stage of design, are of vital importance.

Simulation models, though could be made very accurate, are not cost-effective.. Therefore, they are not adequate at the early stages of design when the design space is very large. Simulations are most valuable however when detailed evaluations

are required at the final stage of design.

Analytical models are cost effective because they are based on efficient solutions to mathematical equations. However, in order for these equations to have a tractable solution, certain simplifying assumptions must be made regarding the structure and behaviour of the model. As a result, analytical models cannot capture all the details that can be built into simulation models. Nevertheless, an analytical model can provide insight into the key factors affecting performance of a proposed system, and determine the sensitivity of performance to parameter changes. Such a model can provide guidance into the overall design of the system and also be useful in the development of more detailed simulation models as the design matures.

1.2 Definition of Parallel Processing:

Parallel processing, in contrast to sequential processing, is a cost-effective means to improve performance through concurrent activities in the computer. Parallel processing, can formally be defined as follows [KAI 84],

Definition : Parallel processing is an efficient form of information processing which emphasizes the exploitation of concurrent events in the computing process of a job. Concurrency implies parallelism, simultaneouty, and pipelining. Parallel events may occur in multiple resources during the same time interval ; simultaneous events may occur at the same instant and pipelined events may occur in overlapped time spans.

Concurrent events in the processing of a job are attainable at various levels. These levels are summarized as follows :

1- Task or procedure level.

2- Interinstruction level.

3- Intrainstruction level.

The first level is conducted among procedures or tasks (program segment). This involves the decomposition of a program into multiple tasks which may be processed concurrently. The second level is to exploit concurrency among multiple instructions. Data dependency analysis is often performed to reveal parallelism among instructions. Vectorization may be desired among scalar operations within DO loops. Finally, in the third level, concurrent operations within each instruction can be exploited. The highest level is often conducted algorithmically and will be discussed further in chapter 2. The lower level is implemented directly by hardware.

Parallel computers are those systems that emphasize parallel processing. Such systems are categorized as follows :

1- Pipeline computers : such systems perform overlapped computations to exploit temporal parallelism. They are more attractive for vector processing, where component operations may be repeated many times.

2- Array Computers : an array processor is a synchronous parallel computer with multiple arithmatic logic unit called processing element (PE). The PEs are synchronized to perform the same function in the same time.

3- Multiprocessor Systems : consist of two or more processors of comparable capabilities that operate asynchronously. All processors share access to common sets of memory modules, IO

channels, and peripheral devices. Each processor has its own local memory and private devices. The entire system is controlled by a single operating system providing interactions between processors and their programs as various levels.

Clearly, pipeline and array computers exploit parallelism at the inter and intra instruction level whereas parallel processing at the task level is adequate in a multiprocessor system. Our primary goal in this work is to develop analytical models for parallel processing in a multiprocessor environment.

1.3 Analytical performance modeling :

Computer systems can be generally characterized as consisting of a set of hardware resources (e.g. processors, channels, disks, etc...) and a set of tasks, or jobs, competing for and accessing those resources. Because there are mutiple jobs competing for a limited number of resources, queues for the resources are inevitable and with these queues come delays. It is, then, natural to model the system by a network of interconnected queues.The purpose of the model is to predict the performance of the system by estimating charactaristics of the resource utilization, the queue lengths, and the queueing delays. Therefore, analytic models of computer systems have been solely based on queueing network (QN) models.

Research in performance modelling methodology has essentially been research in queueing theory. Key advances in computer performance modeling have also been seen as fundamental breakthroughs in queueing theory. Queueing Theory has attained new relevance because of the computer performance modelling application. Furthermore, to a great extent, the direction of

queueing theory has been influenced and driven by this application [HED 84].

Figure1.1 shows the famous QN model of multiprogramming systems, the so called central server model [BUZ 71]. It was introduced to model contention among programs for processors and IO devices. The model is a QN consisting of a J service centers and a population of N active jobs (the multiprogramming level). Service center (SC) 1, represents the processor and service center j; j=2,...,J represents an IO device. Each job is assumed to reside in main memory, and goes through a number of CPU-IO cycles; it executes on the CPU, performs IO on one of the IO devices, and returns to the CPU, repeating this process until it is terminated. The termination of a program and initiation of a new program is represented by a job re-entering service center 1 having completed service from that service center. In order to completely define the model, the following must be specified,

1) The queueing disciplines at each one of the centers.

2) The service requirement of jobs at the centers.

3) The routing probabilities of jobs between centers.

When the above are appropriately defined, the evolution of the network can be represented by a continous time Markov chain (MC), the sate of which is defined by the number of jobs at each SC. However, as N and J increase, the state space of this MC becomes unmanageably large.

For a restricted class of networks called product form networks [KLI 75, CH 81], several computationally efficient analysis algorithms have been developed [CH 81,REI 80]. The

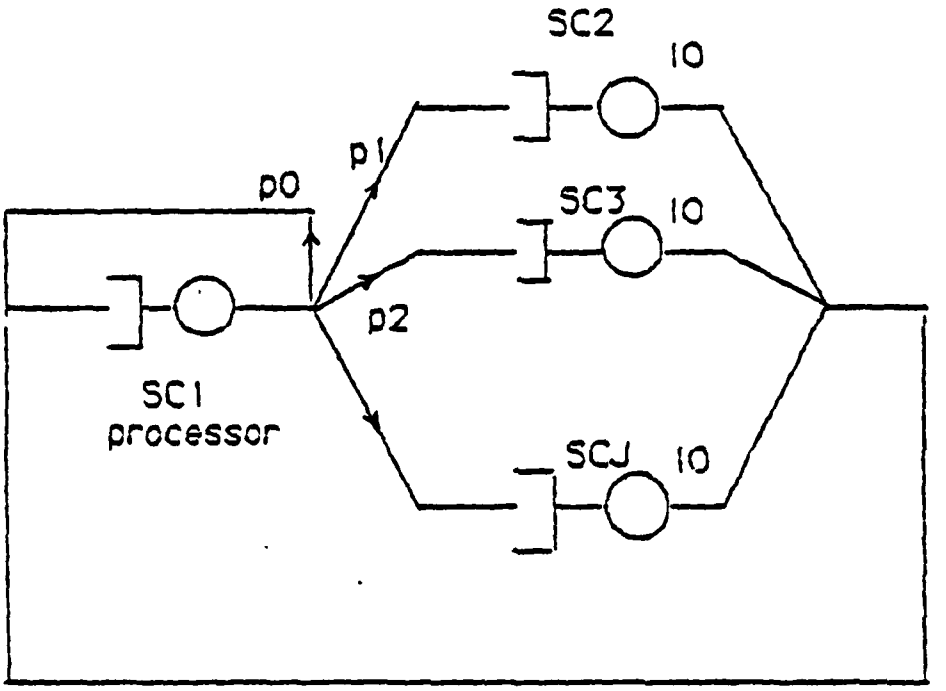


Figure 1.1 The Central Server Model

existence of such algorithms for a broad class of models makes analytic queueing models an attractive tool for applied performance modelling studies of computer systems. The above central server model has been used in several performance prediction studies (e.g. the VM/370 performance predictor [BARD 77,78]).

Product form QNs, however, are not suitable for modelling parallel processing [CH 81, HID 84]. In chapter 2, models of parallel computations will be discussed. A classification of parallel programs based on such models will also be discussed. In chapter 3, current analytical models of parallel processing systems will be briefly described. In chapter 4, models of parallel processing systems using the generalized stochastic petri nets (GSPN) will be presented. In chapters 5, 6, and 7, the analysis techniques for such networks will be developed. Finally in chapter 8, analytical models for systems reliability, maintainability ,availability, and fault diagnosis, using GSPNs, are considered.

CHAPTER 2

MODELS OF PARALLEL PROGRAMS

2.1 Introduction

A parallel program consists of several cooperating concurrent tasks that can be executed in parallel. The terms task and process are intended to mean a self contained portion of a computation that once initiated can be carried out to its completion. The completion of a task is significant in that its occurrence can initiate the execution of another set of tasks.

The problem of defining parallel programs received much attention in the literature. Two approaches were followed, one is to have explicit concurrency, by which the programmer specifies the concurrency using certain language constructs. Conway [con63] proposed a FORK and JOIN statements. FORK spawns a new concurrent process, and JOIN waits for a previously created process to terminate. Dijkstra [DIJ68] proposed a block structure language, which defines concurrent tasks by using the constructs parbegin and parend. For example in the following program segment, the computations for matrices A and B are to be carried out in parallel.

```
begin
 initialize;
 parbegin
    compute matrix A;
    compute matrix B;
 parend
 C = A*B;
end
```

Several general purpose high level languages have incorporated these concepts in their definitions (PL/I, ALGOL-

60, concurrent PASCAL, ADA,..).

The second approach is to have implicit Concurrency. In this case the compiler determines what can be executed in parallel [BAER73].

In section 2, graph models of parallel computations will be described [KAR 66, ADAM70, CER72, BEA77, PET80, MOL81] . These models were developed to facilate the design of parallel programs and deal with the issues of correctnes and efficiency. In section 3, classifications of parallel programs and algorithms will be discussed.

2.2 Models of Parallel Computations

2.2.1 Computation Graphs

Karp, Miller [KAR66], and Adams [ADAM70] have developed models for parallel computations, in which the sequencing control is governed by the flow of data. A directed graph was used to represent the computation. The nodes of the graph represent computation steps, which can range from a single operation to a complex computational task. An edge in the graph can be thought of as a queue of data produced by one node and waiting to be consumed by another. A computation step may be initiated whenever each edge directed into that node of the graph contains the amount of data required for that node to execute properly. The number of computation steps which may be executed at any given time is dynamically determined by the flow of data. Thus unnecessary sequencing constraints may be eliminated.

The properties of the model with which Karp was particularly concerned are : 1- to prove that the model is determinate,i.e, for a given input, the program will yield a unique output

independent of the relative processor speeds; 2- a test to determine whether a given computation will indeed terminate; 3- a procedure for finding the number of performances of each computation step; and 4- the amount of temporary storage required for the data queues associated with the branches (edges) of the graph, together with the conditions for the queue length to remain bounded. The weakness in this model, however, is that data-dependentconditional transfer cannot be taken into account, since the logic at the nodes corresponds to AND-input-AND-output logic, i.e, the computation is started when enough data exist on all input edges, and the output data is placed on all output edges.

The model described by Adams was an attempt to provide a framework within which various classes of computations can be represented. The model has been developed so that computations represented within it will be determinate. The model also deals with data structures , and a hierarchical description of the program. Data structures were treated with generality and include the hardware defined structures such as bits and words, and structures usually defined in a programming language such as arrays, strings, and lists. The hierarchical program description was achieved by being able to treat each node in the graph as representing operation perhaps very complex, and also being able to represent as a graph the suboperations or instructions of which it is constructed. Moreover, data-dependent conditional transfers can be accomodated. This is achieved by dividing the nodes of the graph into two types, computational (r-nodes), which

maps data on the incoming edges to data on the outgoing edges, and computational and logical (s-nodes), which also map edge status as locked or unlocked, an edge with a locked status is treated by successor nodes as empty (contains no data).

2.2.2 Control graphs

Control graphs [CER72, BEA77] are bilogic directed graphs. The arcs contain non-negative number of tokens. Logic expressions are assigned to the set of input arcs and to the set of output arcs for each node in the graph. The expressions are made of "and's" (*) and "or's" (+). Computation is simulated by the movement of tokens from arcs through nodes to arcs. Formally a control graph is defined as follows:

$B=(G,L,Q)$, where $G=(W,U)$ is a directed graph with $W=\{w_1,\ldots,w_n\}$ is the set of nodes, and U is the set of ordered pairs or arcs $u_k=(w_i,w_j)$. There is a unique entry arc with $w_i=\emptyset$. $L=(L^-,L^+)$ being the logic conditions (L^- is the input and L^+ is the output logic. Thus with each node w_i is associated one of the ordered pairs $(*,+),(+,*),(+,+),(*,*)$. If $L^-=*$, w_i is said to be of AND-input logic (respectively OR if $L^-=+$), and similarly if $L^+=*$, w_i is said to be of AND-output logic (respectively OR). Finally $Q=(Q^-,Q^+)$ are the (input,output) token value specifications which map WXU into the set of positive integers N.

The initiation of a computation modeled by w_i can proceed when $L^-=*$ (respectively $L^-=+$) only if for each (at least one) incident arc a there is at least $Q(w_i,a)$ tokens on it. Figure 2 shows an example of a control graph which could be a model for the following segment of a program:

```
Repeat
  node 1; parbegin
            begin action at node 2 .. end
            begin action at node 3 .. end
          parend
Until condition at node 4;
```

The main imputence behind control graph studies is to show that the graphs are terminating properly, i.e.,that they represent correct and terminating programs from the flow of control viewpoint.

2.2.3 Standard Petri Nets

In this section a simple, yet very powerful graph model of behaviour will be presented.

2.2.3.1 Petri Net Structure

Definition 1: A Petri net is a bipartite directed graph

PN = (X,A), where

1- X = P U T is a finite set of nodes with $P=\{p_1,...,p_n\}$ being a set of places, and $T=\{t_1,...,t_m\}$ being a set of transitions, such that $P \cap T = \emptyset$.

2- A = I U O is a finite set of directed arcs with

I: PXT --> B are the input arcs, and

O: TXP --> B are the output arcs, where B = {0,1}.

Inthe sequal, a Petri Net will be referred to by the four tuple PN = (P,T,I,O). And the functions I and O will be called the input and output functions. A place $p_i \in P$ such that $I(p_i,t_j) > 0$ ($O(t_j,p_i) > 0$) is called an input place (output place) of transition $t_j \in T$.

Figure 2.2 shows a Petri Net (PN), with places drawn as circles and transitions drawn as bars.

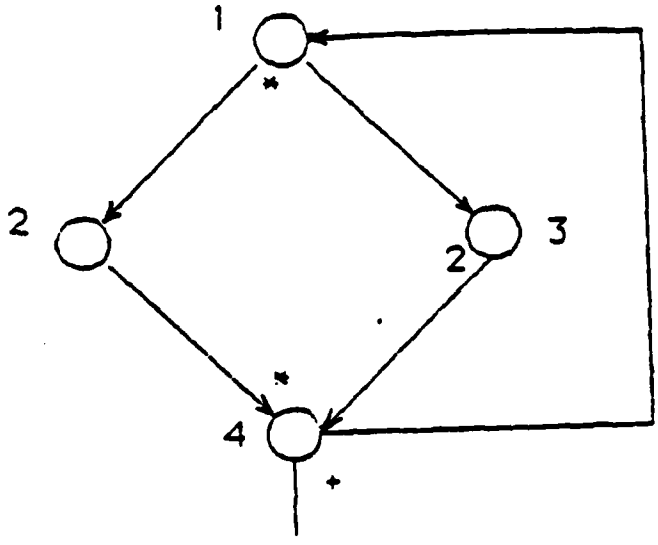


Figure 2.1

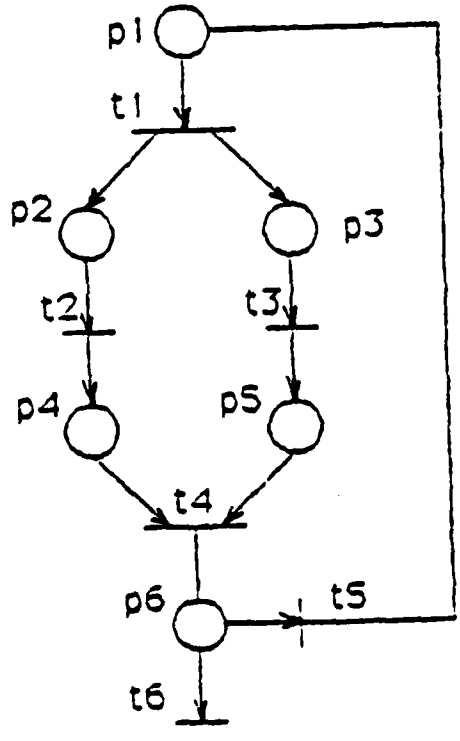


Figure 2.2

$P = \{p_1, p_2, p_3, p_4, p_5, p_6\}$

$T = \{t_1, t_2, t_3, t_4, t_5, t_6\}$

elements of I and O > Ø are

$I(p_1,t_1)$, $I(p_2,t_2)$, $I(p_3,t_3)$,

$I(p_4,t_4)$, $I(p_5,t_4)$, $I(p_6,t_5)$,

$I(p_6,t_6)$, $O(t_1,p_2)$, $O(t_1,p_3)$,

$O(t_2,p_4)$, $O(t_3,p_5)$, $O(t_4,p_6)$,

and $O(t_5,p_1)$.

2.2.3.2 Petri Net Marking

A marking is an assignment of tokens to places of a PN. Tokens can be thought to reside in the places, and the number and position of tokens may change during the execution of a PN. The tokens are used to define the execution of a PN.

Definition 2: A marking M of a Petri Net PN = (P,T,I,O) is a function from the set of places to the nonnegative integers N, i.e., $M: P \dashrightarrow N$.

The marking M can also be defined as an n vector $M = (m_1,.....,m_n)$, where m_i is the number of tokens in p_i, i=1,...,n. The definitions of a marking as a function and as a vector are obviously related by $M(p_i) = m_i$.

2.2.3.3 Execution rules for Petri Nets

The execution of a PN is controlled by the distribution of tokens in the PN places. A place holding one or more tokens is said to be full. A PN executes by firing transitions. A transition is firable (enabled) if all of its input places are full.

Definition 3: A transition $t_j \in T$ in a marked PN = (P,T,I,O) with marking M is enabled if and only if for all $p_i \in P$,

$$I(p_i,t_j) \leq M(p_i)$$

The firing of a transition generates a new marking by removing tokens from the input places and adding tokens to the output places.

Definition 4: A transition t_j in a marked PN with marking M may fire whenever it is enabled. Firing an enabled transition t_j results in a new marking M' defined by

$$M'(p_i) = M(p_i) - I(p_i,t_j) + O(t_j,p_i) \quad , \forall p_i \in P \qquad (2.2.1)$$

M' is said to be immediatly reachable from M.

A more general definition of a PN can be obtained by assigning weights to the input and output directed arcs between places and transition.

Definition 5: a Petri Net is the four tuple, PN = (P,T,I,O), where the input and output functions I and O now are defined as

I: PXT -----> N , and O: TXP -----> N

In this case, following the above definitions for enabling and firing of transitions, a transition is enabled if and only if its input places are full, and each input place holds as many tokens as the weight of the arc linking it to the transition. Moreover, the firing of a transition generates a new marking by removing tokens from the input places and adding tokens to the output places according to the weights of the input and output arcs. In a PN graph a weight label is added to each directed arc (the weight may not be indicated if it is 1).

The state of a PN is defined by its marking. The firing of a transition represents a change in the state of a PN. The state space of a PN with n places is the set of all markings, that is,

N^n. The change in state caused by firing a transition is defined by a change function called the next state function f.

Definition 6: The next state function f: $N^n X T \dashrightarrow N^n$ for a PN = (P,T,I,O) with marking M and for $t_j \in T$ is defined if and only if $M(p_i) \geq I(p_i,t_j)$ for all $p_i \in P$ (i.e.,t_j is enabled in M) . If $f(M,t_j)$ is defined, then $f(M,t_j) = M'$, where M' is defined as in (2.2.1).

Given a PN = (P,T,I,O) and an initial marking M1, we can execute the PN by successive transition firings. Firing an enabled transition t_j in the initial marking produces a new marking M2 = $f(M1,t_j)$. In this new marking we can fire any new enabled transition, say, t_k, resulting in a new marking M3 = $f(M2,t_k)$. This can continue as long as there is at least one enabled transition in each marking. If a marking is reached where no transition is enabled, then no transition can fire, the function f is undefined for all transitions, and the execution must halt.

Two sequences result from the execution of a PN: the sequence of markings (M1,M2,M3,....), and the sequence of transitions which were fired (t_{j1},t_{j2},....). These two sequences are related by the relationship $f(Mk,t_{jk}) = Mk+1$, k = 1,2,3,....

The set of all reachable markings from the initial marking is called the reachability set S.

Definition 7: The reachability set S for a marked PN = (P,T,I,O) with initial marking M1, is the smallest set of markings defined by, 1) $M1 \in S$, 2) if $M' \in S$ and $M'' = f(M',t_j)$ for some $t_j \in T$, then $M'' \in S$.

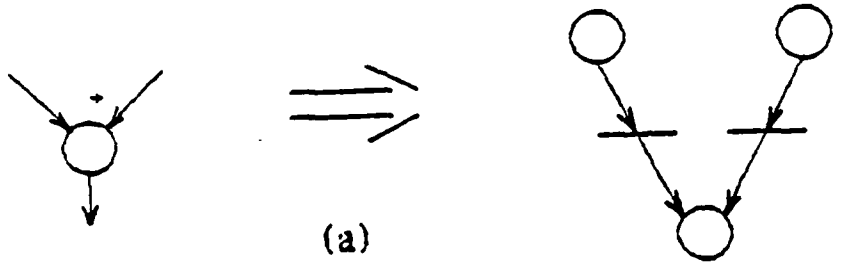
A transition t_i is live, if for all markings $M' \in S$, there

exists an execution sequence which reaches a marking M'' where t_i is firable. A PN is live if all its transitions are live. A PN is said to be k-safe (k-bounded) if a place cannot hold more than k tokens at any time, i.e., if $M(p_i) \leq k$ for all $p_i \in P$ and all $M \in S$. A PN is said to be safe if k = 1.

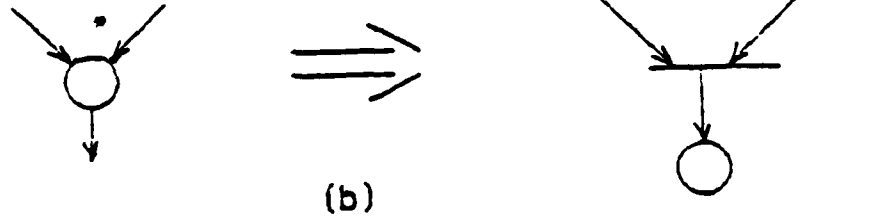
2.2.3.4 Modeling with Petri Nets

Petri nets were used to model various types of systems, where places represent conditions and transitions represent events. Hence a full place shows the holding of a condition, and when all conditions prior to an event are holding, then the event can occur (a transition is enabled). PN models allow all possible states of a system to be examined, so that it can be determined whether sequences of events leading to undesirable conditions exist (e.g. deadlock conditions).

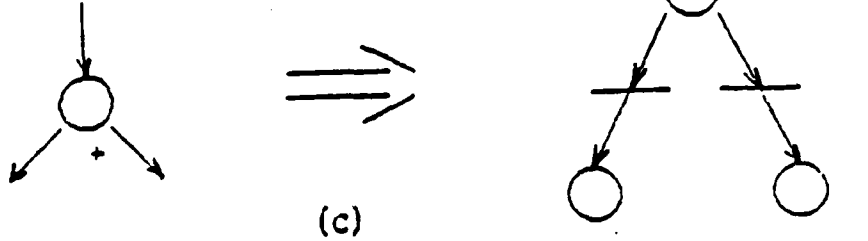
In modeling parallel computations, the firing of transitions in a PN represents the execution of computations, while tokens in places represent the conditions under which computations can take place. A computation sequence follows the execution sequence of transitions. It has been found by Gostlow [GOS71], and Peterson [PET74,80] that the control graphs defined above and PNs computation sequences were in the same theoretical class of formal models. Figure 2.3 shows the mapping of control graphs to PNs.Therefore figure 2.2 is the equivalent PN model of the control graph in figure 2.1 [BEA77,MOL81].



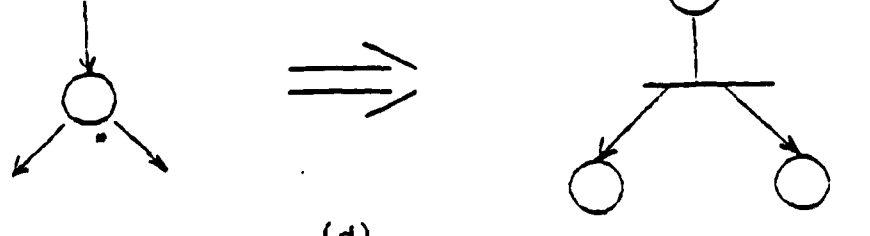
(a)



(b)



(c)



(d)

Figure 2.3

Ramchandani [RAM74] has extended the standard PNs to include a measure of time to what is called Timed PNs. The basic idea of the extension was to simply add a label to each transition which indicated how long that transition takes to fire (which represents the computation time). These time values are fixed (deterministic). Molloy [MOL82], and others, introduced what is called Stochastic PN (SPN), where transition firing times are exponentially distributed random variables. In this case transitions are characterized with their firing rates, which can be marking (or state) dependent. This extension was significant in the sense that it defines a non-deterministic model that can be analyzed by Markov chains (MC). A formal definition of the SPN is thus the following:

SPN=(P,T,I,O,R), where P,T,I, and O are defined as above, and $R=\{r_1,\ldots,r_m\}$ is the set of firing rates associated with transitions.

The problem with analyzing SPNs, however, is that the number of states of the associated MC grows very fast with the dimensions of the graph.

Marsan et al [MAR83], have extended the SPNs to the so called Generalized SPNs (GSPNs). GSPNs are obtained by allowing transitions to belong to two different classes: immediate transitions and timed transitions. Immediate transitions fire in zero time once they are enabled, while timed transitions behave like in SPNs. A formal definition of a GSPN is thus as for SPN, where now the set R contains only m' elements, m' being the number of timed transitions. The significance of this extension is due to the fact that the operating sequence of a system

comprises activities whose duration differ for orders of magnitude. It is then conceivable to model the short activities only from the logical point of view, whereas time is associated with the longer ones. This choice becomes particularly convenient if by doing so the number of states of the associated MC is reduced, hence reducing the solution complexity. Figure 2.4 shows an example of a GSPN (immediate transitions are drawn as double bars). This model is the same as the one in Figure 2.2, except that times for the activities of synchronizing tasks 2 and 3 as well as the conditional transfer at node 4 are neglected. The analysis of the GSPN will be considered in more details later.

2.3. Classifications of Parallel Programs

Using the above models of computations, Herzog et al [HER79] classified the structure of a variety of application programs into four types as follows:

1- Type-1 program structure(figure 2.5 (a)): The program consists of a loop which may be passed several times. This loop consists of a primary task S_0, upon completion of which n independent concurrent tasks are spawned. A new loop may be started if and only if all n tasks are completed. Problems of this type are, algorithms for the solution of linear-algebric or partial differential equations, optimization procedures, simulations including subruns for the purpose of estimating confidence intervals, and problems of picture processing.

2- Type-2 program structure(figure 2.5(b)): Here, the program also consists of a loop. However, the n concurrent tasks

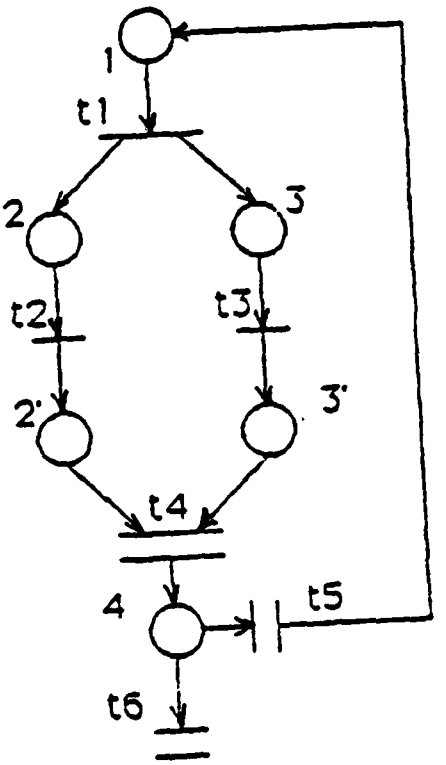


Figure 2.4

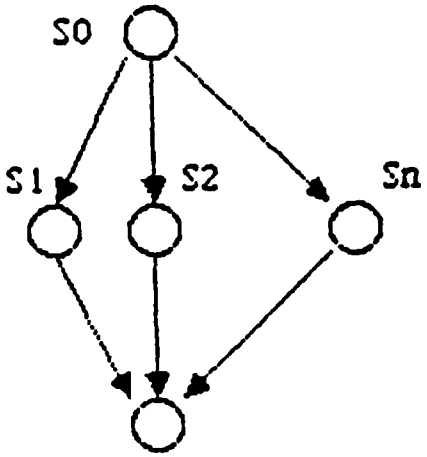
influence each other in some way rather than being completely independent.

Tasks interact at some points called interaction points. These points divide the tasks into stages (subtasks). At the end of each stage a task communicates with some other tasks before the next stage of computation is initiated. Compared to type-1, there are not only global but also local synchronization necessary.

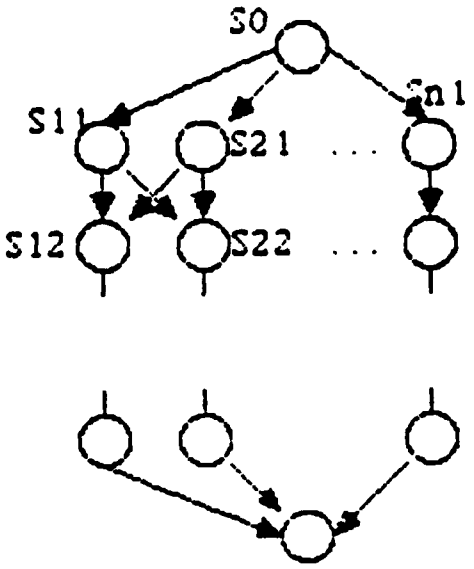
3- Type-3 program structure: Here the degree of parallelism varies rather than being constant. An example is shown in figure 2.5 (c).

4- Type-4 program structure: These program structures consist of completely independent tasks $S_1,....,S_n$ that execute concurrently with the primary task $S_\emptyset$ and terminate independently. Here, no synchronization is necessary since the tasks are completely independent (figure 2.5 (d)).

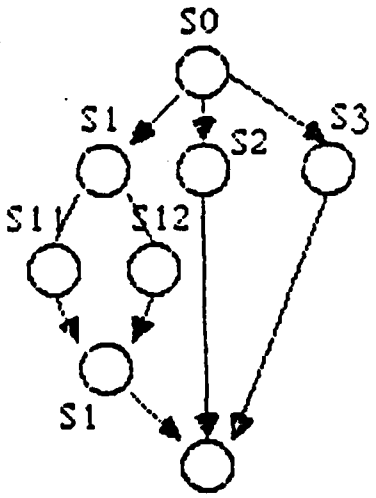
Kung [KUN76, KUN80] classified algorithms for multiprocessors as synchronous and asynchronous. In a synchronized algorithm (type-1 and type-2), the program is decomposed into tasks which are synchronized at interaction points. At these points tasks are blocked while waiting for inputs from others. The loss due to waiting was characterized by a penalty factor defined as follows: Suppose that we want to synchronize K identical tasks, and that the time taken by the ith task is a random variable t_i. Since the tasks are all identical, $t_1,...,t_k$ are identically distributed random variables with mean, say, t. The expected time taken until all of the tasks are completed is the mean T of the random variable $T=\max(t_1,...,t_k)$ rather than t. In general, T is larger



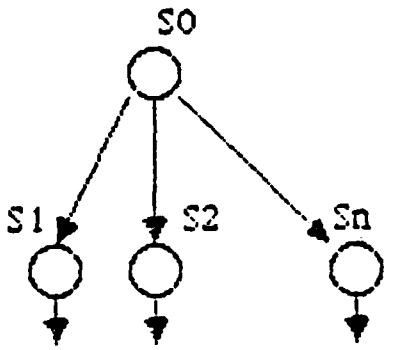
(a) Type-1



(b) Type-2



(c) Type-3



(d) Type-4

Figure 2.5 Classification of Parallel Programs

than t. The penalty factor for synchronizing the k tasks is then defined as the ratio T/t. Clearly if the penalty factor is large, then the performance of the synchronized algorithm is largely degraded. Baudet [BAU76] has observed that, if the t_i's are identical and independent exponentially distributed random variables, then the penalty factor for k tasks is the Kth harmonic number H_k. Note that H_k grows like ln k as k increases. Hence synchronized algorithms should be used when there are only few tasks to be synchronized. Futhermore, the execution time of the needed synchronization primitives is usually non-negligible. Thus, it is not always advantageous to create as many concurrent tasks as possible according to the maximal decomposition of a problem.

Asynchronous parallel algorithms (type-4 program structure), consist several concurrent asynchronous tasks. Communication between these tasks is acheived through a set of global variables or shared data. The main characteristic of these tasks is that they never wait for the completion of others at any time, but continue or terminate according to whatever information is currently contained in the global variables. However, to insure logic correctness, the operation on global variables are programmed as critical sections. This asynchronous behaviour leads to serious issues regarding the correctness and efficiency of an algorithm. The correctness issue arises because during the execution of algorithm operations from different tasks may interleave in an unpredictable manner. The efficiency arises because any synchronization introduced for correctness reasons takes extra time and also reduces concurrency. Kung examined

various techniques for dealing with these issues, and also showed examples for synchronous as well as asynchronous implementations of zero searching and iterative algorithms.

CHAPTER 3

CURRENT MODELS OF PARALLEL PROCESSING SYSTEMS

In this chapter, current analytical models proposed in the literature for parallel processing systems will be discussed [MAE 76,PET 75,PRI 75,TOW 75,TOW 78,BAR 79,HEI 82,HEI 83,TOM 84].

3.1 Models for CPU:IO Overlap.

The models developed in [MAE 76,PET 75,PRI 75,TOW 75,TOW 78] were primarily intended to model CPU:IO overlap using double or multiple buffering. This means that a program issues two or more concurrent requests on distinct system resources. In the following paragraph, we describe briefly the most recent of such models developed by Towsley [TOW 78].

The model developed in [TOW 78] is based on the central server QN model. The assumption normally made in this model is that a job alternates between CPU and IO activities. The job may be thought of as repeating cycles, where each cycle consists of two tasks : a task requiring the use of CPU followed by one requiring the use of an IO. In an overlap system (figure 3.1), a cycle may consist of three tasks; CPU_1 followed by the concurrent tasks CPU_2 and IO. The approximate aggregation technique known in QNs as Norton's Theorem was used to obtain a network with two queues; the CPU queue and an aggregate IO queue. The aggregate network under the overlaped job cycle in figure 1 using the exact recursive analysis of Markov models of two queue networks developed by Herzog, Woo, and Chandy [HER 75].

The accuracy and validity of the model have been verified against detailed simulations. This model can also be extended to

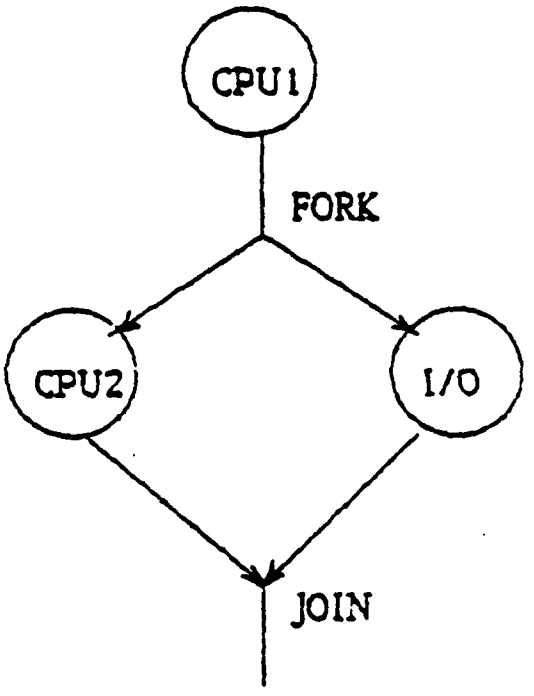


Figure 3.1

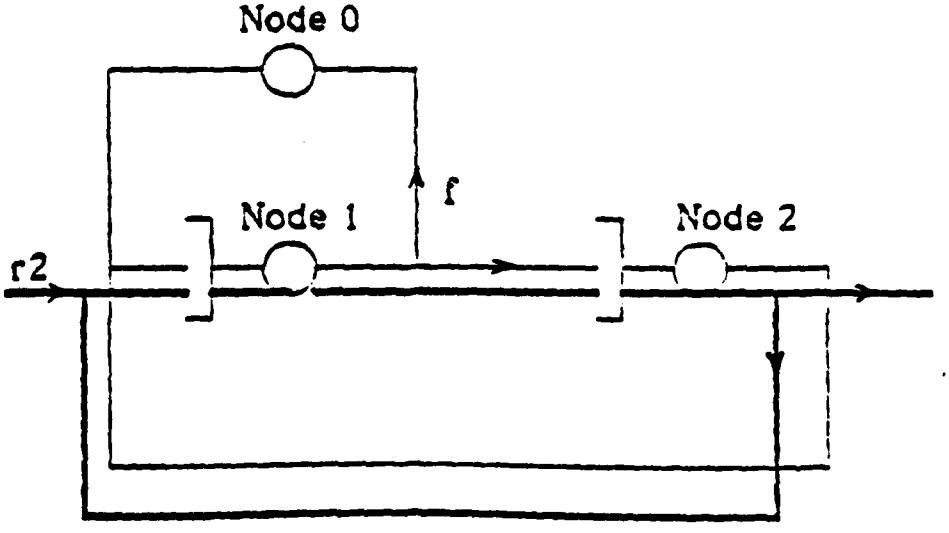


Figure 3.2

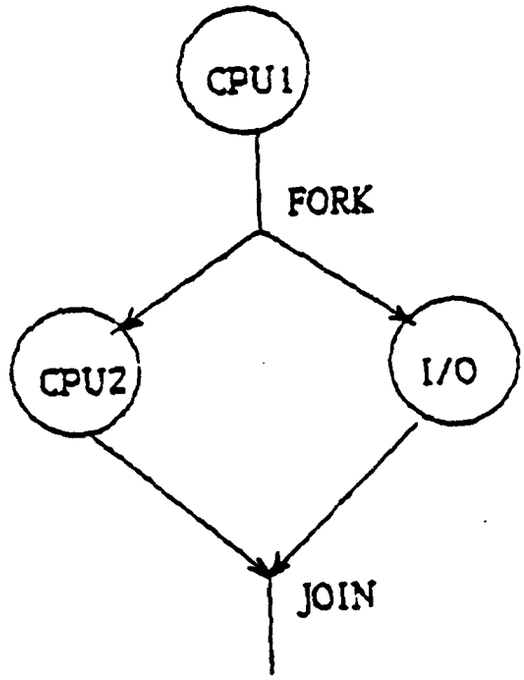


Figure 3.1

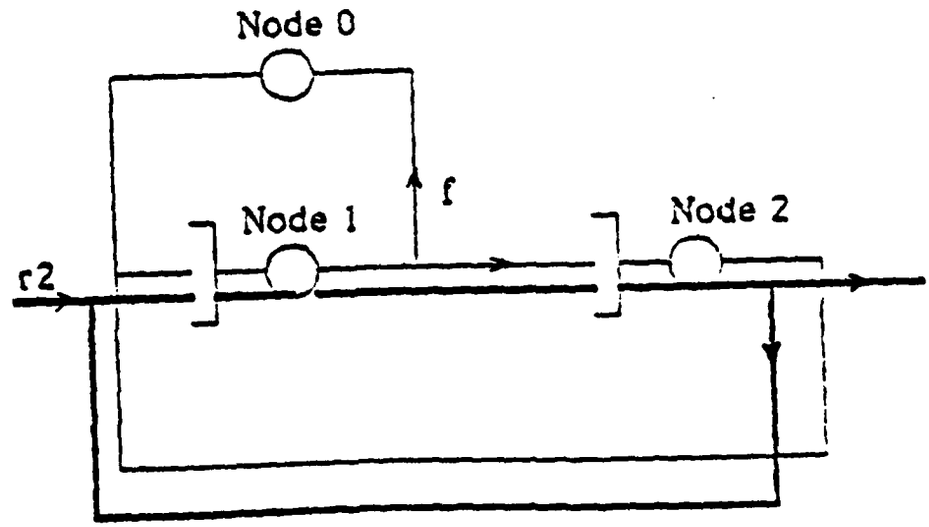


Figure 3.2

model overlap in CPU:CPU and IO:IO activities. However, such concurrent tasks use only one resource each before they synchronize and merge.

Models of systems with programs defined as a set of concurrent tasks, where each task requires multiple accesses to many different system resources before it either terminates independently or communicates with other tasks, have been developed in [HED 82,HED 83,TOM 84]. The remaining of this chapter will be devoted to the description of such models.

3.2 Models for Asynchronous Tasks:

The model developed in [HED 82] does not account for any synchronyzation between tasks. It assumes a system workload consisting of a set of statistically identical jobs. Each job consists of a primary task (labeled 1) and zero or more statistically identical secondary tasks (labeled 2) . The secondary tasks are spawned by the primary task sometime during its execution and execute concurrently with it, competing for systems resources. A secondary task is otherwise assumed to run and terminate independently. An approximate multi-chain QN model was developed with two chains; one closed and the other open. The closed chain models the execution of primary tasks in the system and a specially defined node (node Ø) was defined such that a secondary task is spawned whenever a primary task enters this node. The open chain which shares the same systems resources with the closed chain, models the execution of secondary tasks which compete for the system resources with the primary task, and leave the network when they terminate. The arrival rate of the open chain is set equal to the throughput of primary tasks,at node Ø

,of the closed chain. The arrival process of primary tasks at node Ø is approximately assumed to be Poisson which is independent of the state of the network. Since the throughput of the closed chain is itself a nonlinear function of the arrival rate of the open chain, a closed form solution is not available. An iterative algorithm was used to solve this nonlinear equation, and the conditions for its convergence were given. The accuracy of the approximation was studied through comparison with simulation.

To illustrate the above model, consider the two-chain QN chain shown in figure 3.2.

Let N =number of primary tasks,

r_2=arrival rate of the open chain

r_{ij}=throughput of task j at queue i; i=Ø,1,2, and j=1,2,.

v_{i2}=mean number of visits a secondary task makes to queue i (open chain)

u_{i2}=utilization of server at queue i due to secondary tasks.

Q_{ij}=mean queuing time of task j at queue i.

L_{ij}=mean queue length of task at queue i.

S_{ij}=mean service time of task j at server i.

f=probability that a primary task will go to node Ø.

P_2=probability that a secondary task return to Q_1 for some more service.

At steady state,

$v_{12}=1+ P_2\ v_{12}$, $v_{12}=1/(1-P_2)$

Then , $r_{12} = r_2 v_{12}$

$u_{12} = r_2 v_{12} s_{12}$, and

$u_{22} = r_2 v_{12} s_{22}$

Then using the MVA algorithm for mixed QNs [REI 80], we have,

$Q_{i1}(N) = s_{i1}[1+ L_{11}(N-1)]/(1-u_{12})$

$Q_{21}(N) = s_{21}[1+ L_{21}(N-1)]/(1-u_{22})$

$r_{11} = N/[Q_{11}(N)+(1-f)Q_{21}(N)]$

$r_{01} = f\ r_{11}$, and $r_{21} = (1-f) r_{11}$

$L_{i1}(N) = r_{i1}(N)\ Q_{i1}(N)$; $i=1,2$

From the above,

$$r_2(N) = r_{01} = fN/\{[(s_{11}/(1-r_2(N)v_{12}\ s_{12}))(1+L_{11}(N-1)]+ [((1-f)s_{22}/(1-r_2 v_{12} s_{22}))(1+L_{21}(N-1))]\}$$

The above nonlinear equation is solved iteratively for r_2, such that $r_2 v_{i2} s_{i2} < 1$; $i=1,2$

If this condition is not satisfied, then a stable solution cannot be obtained, which means that primary tasks are able to generate more secondary tasks than the system can handle, and the mean queue length of secondary tasks at at least one queue will be infinite.

3.3 A Model for Synchronous Tasks:

In [HED 83], another model was developed for synchronous tasks. It assumes a workload consisting of a set of statistically identical jobs. Each job consists of a primary task and a fixed number of synchronous concurrent secondary tasks. The system again consists of a finite number of servers including processors and IO devices, a particular pseudo server labeled 0, and a finite number N of jobs. Each primary task of a job executes on a sequence of servers and whenever it enters server 0, it splits

into a fixed number (>=2) of secondary tasks. Each primary task is the parent of its secondary tasks and later are said to be siblings. Each secondary task executes on a sequence of servers. A secondary task is considered complete upon entering node Ø. At node Ø, the secondary task must wait until all of its siblings have completed execution, at which time the primary task becomes active again and the process is repeated. Syncronization between secondary tasks is achieved by requiring all siblings to complete execution before the job can continue processing. Two approximations were proposed to come up with a tractable solution for the above model. Both approximations are based on solving a set of related multi-chain closed QNs.

The first method is based on decomposition approximation, following the decomposition approach of Courtois [COU 77]. For purposes of illustration, we restrict our discussion to the case in which a primary task subdivides into two secondary tasks. Each of the N jobs may either have its primary task (labeled Ø) or have one or both of its secondary tasks (labeled 1 and 2) active. At time t , let $a_i(t)$ denote the number of active,i.e. executing, tasks of type i, and let $w_i(t)$ denote the number of i secondary tasks which have completed execution and are waiting for their respective sibling to complete ; i=1,2. Let $a(t)=(a_\emptyset(t),a_1(t),a_2(t))$ and $w(t)=(w_1(t),w_2(t))$. Then at time t ,$N=a_\emptyset(t)+a_i(t)+w_i(t)$ for i=1,2.

If the primary and secondary tasks are relatively long in the sense that many servers are visited before the task, then changes in a(t) will occur frequently as compared to changes in the queue

lengths of tasks at the servers. In this case, it is reasonable to assume that queue length distributions converge to steady state distributions prior to the next change in a(t). Therefore, for every feasible multiprogramming level(mpl) $a=(a_0,a_1,a_2)$, a closed three-chain QN, with population a_0,a_1,a_2 respectively is solved for the performance parameters (throughputs, utilization , and mean queue lengths). Let $r_i(a)$ denote the throughput of type i tasks at node 0 when the mpl is a. The process $\{a(t), t>=0\}$ is then modeled as a finite state Markov chain with state space E and transition rate matrix Q. The state space E is a subset of $\{0<a_0<N,\ 0<a_1<N-a_0, 0<a_2<N-a_0\}$. The exact definition of E however is quite complex. Another assumption was made to facilitate the state space definition by keeping track of only the mpl without specifying the identity of the waiting tasks. Let $p_i(a)$ be the probability that a type i secondary task that has just completed finds that its sibling has also completed, when the mpl is a. These probabilities are defined as follows,

$$p_1(a) = \begin{cases} w_2/a_1 & \text{if } a_1>0 \\ 0 & \text{if } a_1=0 \end{cases}$$

$$p_2(a) = \begin{cases} w_1/a_2 & \text{if } a_2>0 \\ 0 & \text{if } a_2=0 \end{cases}$$

Then the state space E is given by,

$E=\{ a: 0\leq a_0\leq N , 0\leq a_1\leq N-a_0, 0\leq a_2\leq N-a_0, N\leq a_0+a_1+a_2\}$, and the off diagonal elements of the rate transition matrix Q (q(a,b)) are listed below ($q(a,a) = -\sum_{a=/=b} q(a,b)$).

b	a	transition explanation
(a0-1,a1+1,a2+1)	r0(a)	task 0 completed, tasks 1,2 spawned
(a0,a1-1,a2)	r1(a)(1-p1(a))	task 1 completed, sibling active
(a0,a1,a2-1)	r2(a)(1-p2(a))	task 2 completed, sibling active
(a0+1,a1-1,a2)	r1(a) p1(a)	task 1 completed, sibling waiting
(a0+1,a1,a2-1)	r2(a) p2(a)	task 2 completed, sibling waiting

The stationary distribution P of this Markov chain can be obtained from solving PQ=0, such that the elements of P sums to one. The global performance parameters can then be obtained. e.g., the job throughput r is given by,

$$r = \sum_{a \in E} P(a)\ r_0(a)$$

The second approximate method proposed is based on the method of complementary delays and is similar to the one proposed in [IAC 81] for simultaneous resource possession. It consists of iteratively solving a sequence of product form (pf) QNs. The synchronization delay experienced by a task was modeled by an infinite server queue. The mean synchronization delays are estimated by assuming the tasks response times to the independent, exponentially distributed random variables. The mean tasks response times are obtained from the solution to the pf QN at the previous iteration. However, the decomposition approximation was found to be clearly superior to the above method. The former was found to be remarkably accurate, when compared to simulation results, predicting throughputs and device utilizations with a mean relative error of one percent.

2.4 A Model for a Task System:

Thomasian et Al [TOM 83] developed a QN model based on the above decomposition approximation. The model assumes that there exists only one active job in the system (monoprogramming), which is defined by a set of tasks $T=\{T_1,T_2,\ldots,T_n\}$ with a partial order defined on T specifying deterministic precedence constraints. Only a directed acyclic graphs were considered. from such a graph, a MC is constructed, the state of which defines the active task combination. A closed multi-class QN is solved for each state of the MC.

CHAPTER 4

MODELING PARALLEL PROCESSING SYSTEMS USING THE GENERALIZED STOCHASTIC PETRI NETS

4.1 Introduction

In the previous chapter current models for parallel processing were discussed. Such models were based on analytical queuing networks (QNs). And due to the fact that analytical QN models become intractable when modeling parallel operations, several approximate models were developed. Even though the accuracy of such models were found to be adequate, they were restricted to a workload consisting of statistically identical jobs with one type of parallelism or another. For example, the model in section 3.2 assumes only asynchronous concurrent tasks. In section 3.3, the model was developed for jobs with fixed number of synchronous tasks. And in section 3.4, the model is restricted to one active job consisting of a fixed number of tasks with deterministic precedence relation.

From the classification of parallel programs given in chapter 2, it is clear that the active jobs in the system may consist of both synchronous and asynchronous tasks. Moreover, a probabilistic model is needed to represent a wide variety of active job structures.

In this chapter, a different analytical modeling tool, namely, the Generalized Stochastic Petri Nets (GSPNs), is used to model activities in a parallel processing system. Such activities include parallel operations, synchronization, contention for resources, and queuing.

GSPNs are a very versatile modeling tool. The probabilistic nature of GSPNs allows systems operations to be described in a high level of abstraction. As indicated in chapter 2, Petri Nets (PNs) are very powerful in modeling parallel activities and synchronization. A distinctive feature of GSPNs, with respect to standard PNs models, is their isomorphism with markovian models, which allows evaluation of the performance of a system. However, GSPN models eliminate a major difficulty in the direct construction of a Markov chain, that is, its state space definition. Also GSPN models retain much of the characteristics of the system, therfore they provide greater insight into the various system activities.

In section 2, a more detailed description of GSPNs will be given. The analysis of GSPNs, however, will be deferred to the next chapter. In section 3, modeling parallel processing systems using GSPNs will be considered. And in section 4, an approximate hierarchical model for large scale systems, using both tractable QNs and GSPNs, will be presented.

4.2 Description of GSPNs

Recall the definition of a Stochastic Petri Net (SPN) [MOL81], which consists of a set of places P, a set of transitions T, the input output functions I and O, and the set of transition firing rates R.

$$SPN = (P,T,I,O,R),$$

where, $P = \{p1,p2,\ldots,pn\}$, $T = \{t1,t2,\ldots,tm\}$,

$I:PXT \dashrightarrow N$, $O:TXP \dashrightarrow N$, and $R = \{r1,r2,\ldots,rm\}$

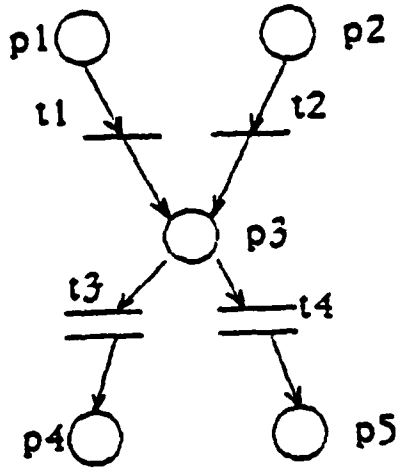
For a given initial marking M1, the reachability set S can be

obtained using the same analysis performed on standard PNs.

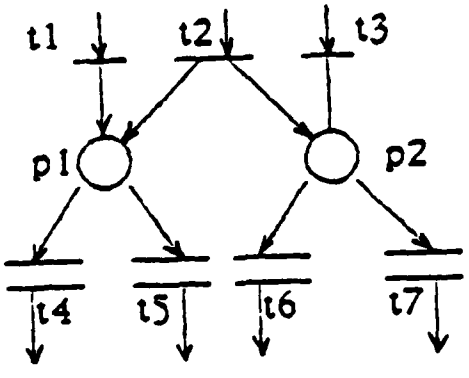
Once enabled, a transition ti ϵ T takes an exponentially distributed random time, with rate r_i, to fire. In a GSPN transition firing rates can be infinite. Therefore the set T is partitioned into a set of timed transitions with finite firing rates defined in the set R, and a set of immediate transitions. Clearly, for any marking in S at which several timed transitions, and one immediate transition, are enabled, the immediate transition fires with probability one. However, if several immediate transitions are simultaneously enabled at a marking, then it is necessary to define a probability density function on the set of enabled immediate transitions according to which the firing immediate transition is selected. This is defined more precisely as follows,

Definition 1 : Let Ti(M) = {ti1,ti2,...,tik} be the set of all enabled immediate transitions at a marking M ϵ S. If k > 1, then a probability distribution, called the switching distribution $P_M(tij)$, j=1,..,k, with $\sum_{j=1}^{k} P_M(tij) = 1$, is defined on the set Ti(M) such that transition tij fires with probability $P_M(tij)$. This set of immediate transitions together with the associated switching distribution is called a random switch.

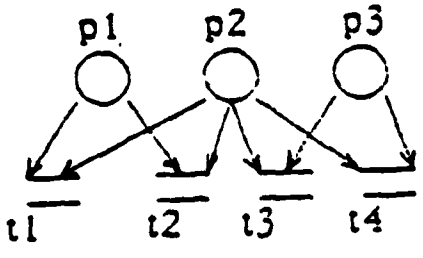
The reachability set S of a GSPN is a subset of the reachability set of the associated PN, because precedence rules introduced with the immediate transitions do not allow some states to be reached. For example, figure 4.1 shows a GSPN where {t1,t2} are timed transitions , and {t3,t4} are immediate transitions. Clearly a marking with two tokens in place p3



(a)



(b)



(c)

Figure 4.1

cannot be reached. Note also that a switching distribution must be defined on transitions t3 and t4 since they are simultaneously enabled when there is a token in p3.

A crucial aspect of the definition of a GSPN is the identification of all random switches and the correct definition of the switching distribution. This distribution, however, cannot be clearly specified when the set Ti contains independent transitions. Since it represents possibly unclear relation between fast independent events in a system. In the following we restrict our discussion on immediate transitions to address this issue.

Definition 2: a set of transitions are said to be independent if and only if they do not share any input place. Let Pin = { pk / I(pk,ti) > 0 } be the set of input places of transition ti $\in$ T. Then transitions ti,tj $\in$ T are independent if and only if, Pin(ti) $\cap$ Pin(tj) = $\emptyset$. Otherwise they are said to be dependent.

For example consider the portion of a GSPN shown in figure 4.1 (b). Assume t1 fires first, so that a token moves to place p1, thus enabling the immediate transitions t4 and t5. Since these transitions are dependent, the switching distribution can be easily specified because it depends on some local behaviour of the system. Let P(t4) and P(t5) be the switching distribution defined on {t4,t5}. Similarly, if t2 fires first, let P(t6) and P(t7) be the distribution defined on {t6,t7}. Now if t3 fires first, a token is placed in both p1 and p2, thus enabling the four immediate transitions t4, t5, t6, and t7. However, since transitions in {t4,t5} and {t6,t7} are independent , the switching distribution on {t4,t5,t6,t7} accounts for possibly

unclear relation between two separate parts of ths system.

It is implicitly assumed in the above definition of a random switch that no two transitions can fire simultaneously even if they are independent. In the sequal we relaxe this assumption to find a more general definition of the switching disribution.

Definition 3: a set of enabled transitions are said to be mutually exclusive , i.e., only one of them can fire , if they are dependent.

Definition 4: for each set Tj = {tj1,tj2,..,tjr} of mutually exclusive immediate transitions, define a switching probability distribution Pd(tjk), k = 1,2,..r, such that $\sum_{k=1}^{r}$ Pd(tjk) = 1.

For example figure 4.1 (c) shows a set of dependent transitions {t1,t2,t3,t4} with a common input place p2. In a marking with a token in both p1 and p2, t1 and t2 are mutually exclusive and a switching distribution must be specified on the subset {t1,t2}. Similarly, for a marking where p1, p2, and p3 are full, a switching distribution must be defined on the set {t1,t2,t3,t4}.

Now let H(M) be the set of all enabled immediate transitions at a marking M. H(M) can be partitioned into several subsets of mutually exclusive transitions Qi(M), i=1,..,z, such that for any tu $\notin$ Qi(M) and tv $\in$ Qj(M), i$\neq$j, tu and tv are independent. Let Pd^{Qi}(tij) be the local switching distribution defined on the set of mutually exclusive transitions Qi(M). Assuming that independent transitions can fire simultaneously, let E^M = {E1,E2,..,EJ} be the event space at marking M, where Ei, i=1,...,z, are the events when only one transition in Qi(M) fires, Ei, i=z+1,...,z!/(2!*(z-2)!) are the events when exactly

two transitions fire simultaneously, and EJ, J=(2^z-1), is the event when z transitions fire. Also let P(Ei) be the probability of event Ei, i=1,...,J, such that $\sum_{i=1}^{J} P(Ei) = 1$. These probabilities can be defined by the system analyst if the relations between the parts of the system represented by the sets of transitions Qi(M) is clear. If this is not possible, the above events can be assumed to be equally likely , and in this case

$$P(Ei) = 1/(2^z-1) \quad , i=1,.....,(2^z-1)$$

The global switching distribution of the set H(M) can then be defined as follows,

$$P_M(ti1,ti2,...,tik) = P(Ek) \cdot Pd^{Qj1}(ti1) \cdot Pd^{Qj2}(ti2) \cdot \cdot \cdot \cdot Pd^{Qjk}(tik) \quad , \quad k=1,....,(2^z-1)$$

Where $P_M(ti1,ti2,...,tik)$ is the probability that the set of transitions {ti1,ti2,..,tik} will simultaneously fire, Ek is the event that k transitions one from each Qjs, s=1,..,k, will fire, and $Pd^{Qij}(tij)$ is the probability that transition tij ϵ Qij will fire.

4.3 Modeling Parallel Processing Systems Using GSPNs

Because of the nature of parallel processing systems, a model of such systems has to handle such phenomena as contention for multiple resources, queueing, parallel and sequential operations. As demonstrated earlier, unlike petri nets, analytical queueing network models are not adequate for modeling parallel operations. However, they are very powerful in modeling contention for resources. In this section, we demonstrate by

examples that GSPNs can also be used to model contention for multiple resources.

Molloy [MOL81] addressed queuing issues in Stochastic Petri Nets (SPNs). Where places are viewed as queues, and transitions model arrival and departure events. For example consider the SPN shown in figure 4.2 (a). This SPN represents a service center with a stream of customers arriving with an exponentially distributed inter-arrival time with a state dependent rate r1(m), and exponentially distributed service time with state dependent rate r2(m), where m is the number of customers in the center (number of tokens in p1).

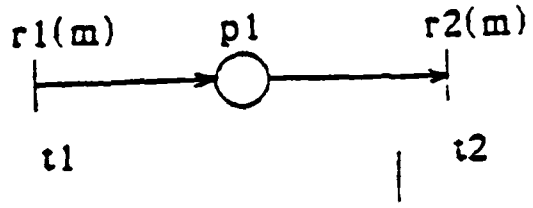
If r1(m) and r2(m) are constants, i.e., they do not depend on m, then the SPN represents an M/M/1 queue. for an M/M/k queue, r1(m) is again constant, and r2(m) is given by,

$$r2(m) = m \cdot r2 \quad , 0 < m \leq k$$
$$= k \cdot r2 \quad , m > k$$

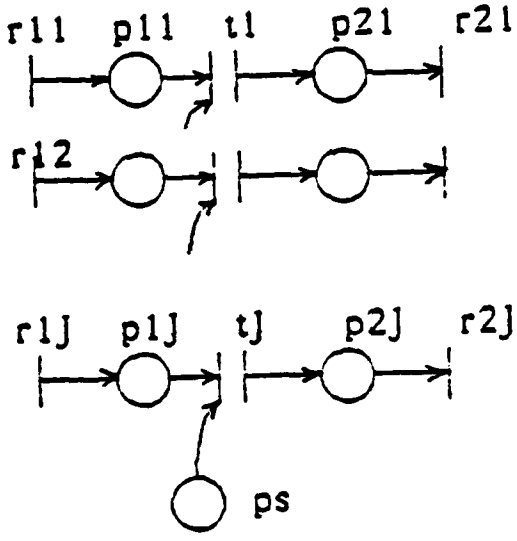
Due to the memoryless property of the exponential distribution, and since we have only one class of customer, the above is adequate for both first-come first-served (FCFS) and last-come first-served (LCFS) queuing disciplines. If the queuing discipline is processor sharing (PS), then r2(m) will be given by,

$$r2(m) = m \cdot r2 \quad , 0 < m \leq k$$
$$= k.r2/m \quad , m > k$$

It is not possible, however, to model service centers with mutiple classes of customers and fixed priority or FCFS queuing



(a)



(b)

Figure 4.2 (a) and (b)

disciplines by SPNs. Since in this case we must take into account the order of customers waiting for service. As shown in the previous Chapter, this class of service centers is very important in models of parallel processing systems. In the sequel we will show that this class of service centers can be modeled by GSPNs.

Figure 4.2 (b) shows a GSPN that represents a service center with J classes of customers. A token in pli, i=1,..,J, represents a class i customer has arrived and is waiting for service. A token in p2i, i=1,...,J, represents that a type i customer is being served. And tokens in ps represent the number of available servers or resources. Again the inter-arrival (service) time of class i customers is exponentially distributed with rate rli (r2i).

The queuing discipline is assumed to be of a fixed priority, with class 1 customers have the highest priority and class J customers have the lowest priority. When a resource becomes available, and if there are customers waiting, several immediate transitions are simultaneously enabled. And their switching distribution is defined such that the transition that corresponds to the highest priority class will fire with probability one.

Figure 4.2 (c) shows a GSPN model of a service center with J customer classes and FCFS queuing discipline. The queue portion of the model is divided into s stages where customers are ordered according to their arrival. It is assumed in this model that the maximum number of customers that can exist in the center is (s+k), where k is the number of available resources (number of tokens in ps).

The following example demonstrates the ability of GSPNsto

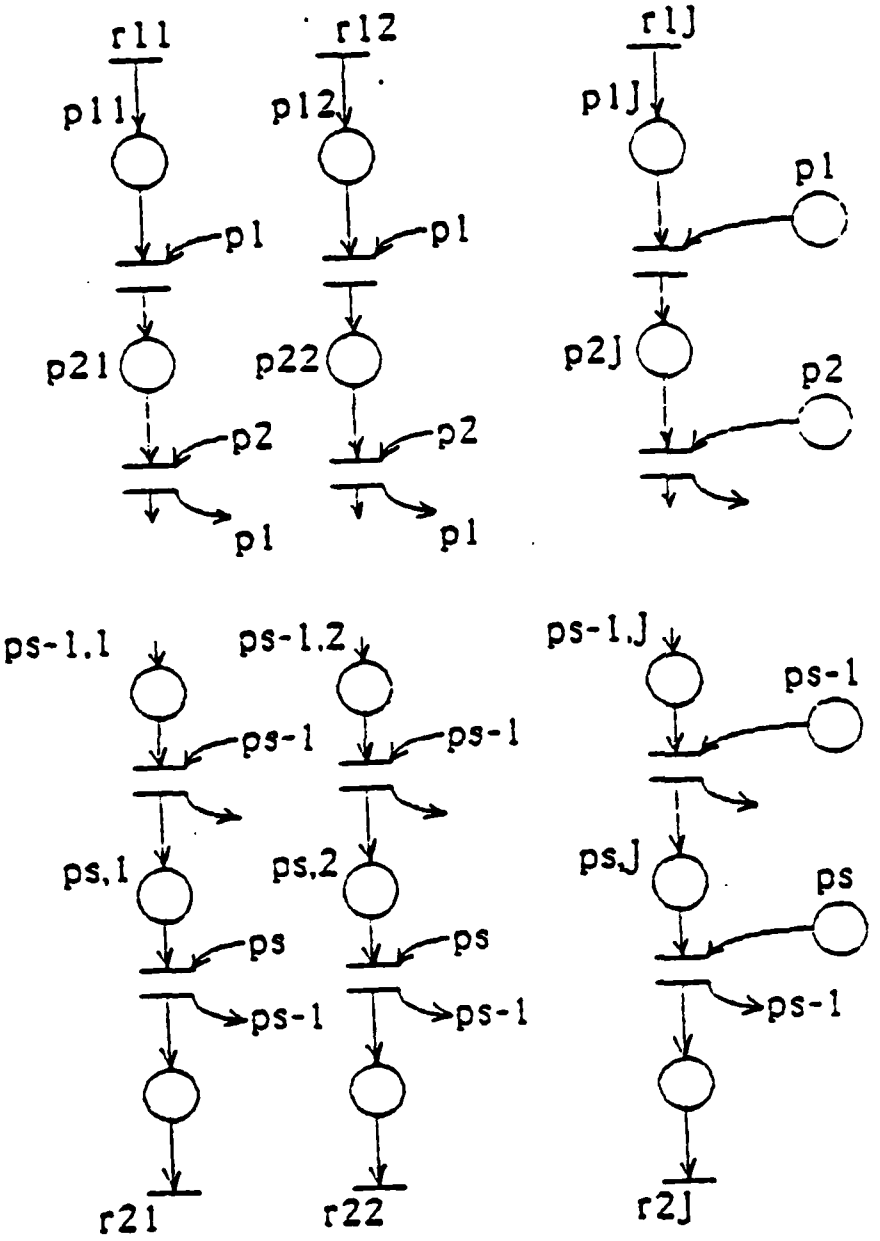

r11
p11
p1
p21
p2
p1
r12
p12
p1
p22
p2
p1
r1J
p1J
p1
p2J
p2
ps-1.1
ps-1
ps,1
ps
ps-1
r21
ps-1.2
ps-1
ps,2
ps
ps-1
r22
ps-1.J
ps-1
psJ
ps
ps-1
r2J

Figure 4.2 (c)

model sequential and synchronized parallel operation, and contention for multiple resources.

Example 4.1 : Consider a system consisting of two service centers. Service center one (SC1) consists of an infinite number of servers (e.g. IS queue). Service center two (SC2) consists of two resources with FCFS scheduling policy. Jobs in the system consist of either a sequential task (labeled 1), or a sequential task followed by two synchronous parallel tasks (labeled 2 and 3). The GSPN shown in figure 4.3 models the activities in such system when there is only one job in the system at a time. A token in places 1,2 and 3 indicates that a type i task ; i=1,2,3 is being serviced at SC1. Also a token in places 10, 11 and 12 indicates that a type i task is being serviced at SC2 and a token in places 14,15 and 16 indicates that task i has completed. When a job is completed we assume that a similar one immediatly enters the system. The rates r_i of timed transitions t_i ; i=1,2,...,6 , are given by,

$$r_i = m_i/s_{i1} \quad ; i=1,2,3$$

$$= 1/s_{k2} \quad ; k=1,2,3 \ , \ i=4,5,6 \ ,\text{respectively}$$

where m_i is the number of tokens in place i, and s_{ij} = mean service time of type i task at SCj ; i=1,2,3 , j=1,2.

Also, the branching probabilities (of immediate transitions) are defined as follows,

P_{ijk} = probability that a type i task will visit SCk upon completing service at SCj , i=1,2,3 , j,k=1,2, and

P_{ij0} = probability that a type i task will terminate upon visiting SCj ; i=1,2,3 , j=1,2.

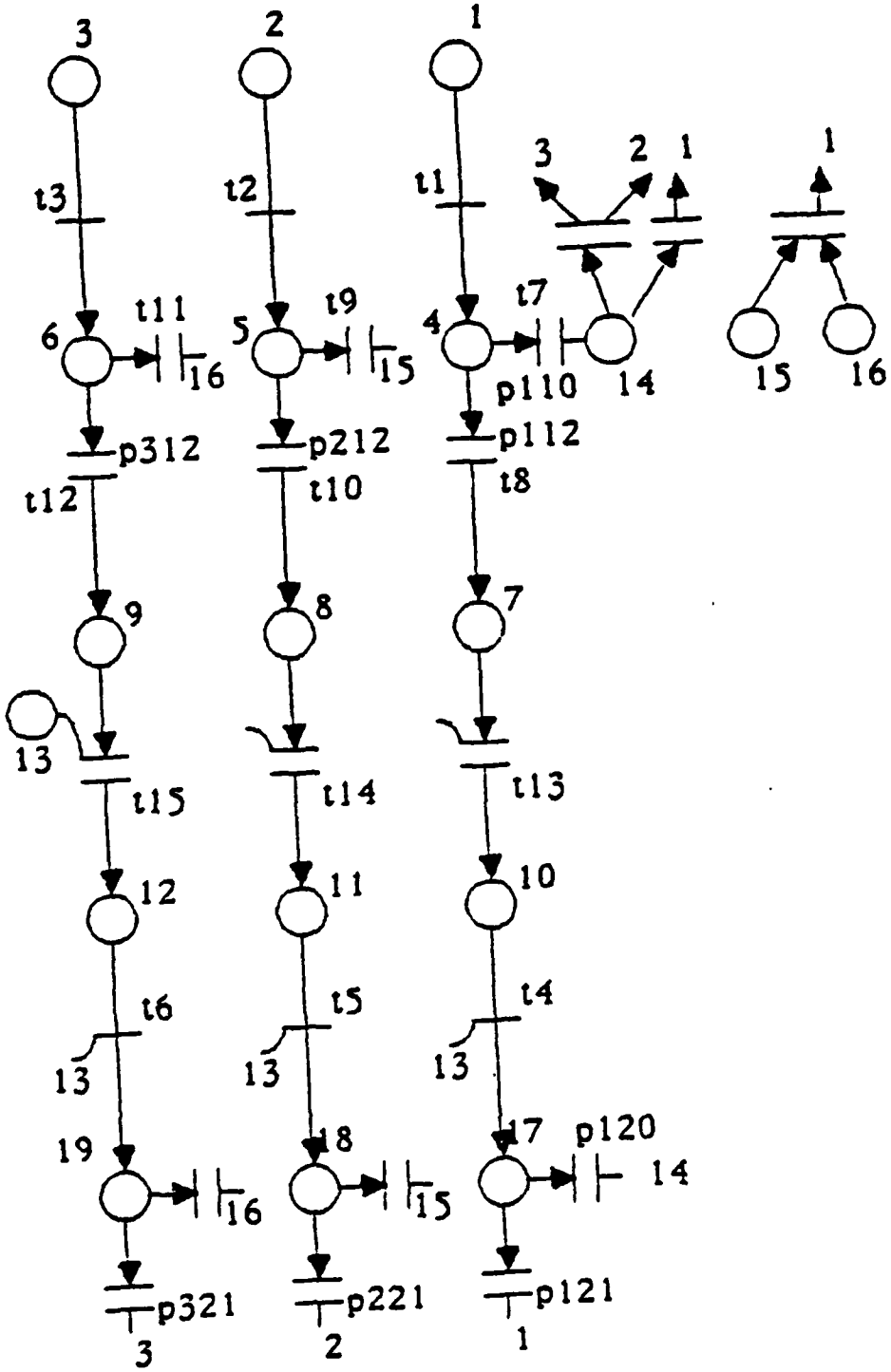

3
2
1
t3
t2
t1
3 2 1
1
t11
t9
t7
6
5
4
16
15
p110
14
15
16
p312
p212
p112
t12
t10
t8
9
8
7
13
t15
t14
t13
12
11
10
t6
t5
t4
13
13
13
19
18
17
p120
16
15
14
p321
p221
p121
3
2
1

Figure 4.3

Also, p' is the probability that a type 1 task, upon terminating, will spawn two synchronous tasks.

Transition t16 models the synchronization operation between the two parallel tasks and is enabled when both tasks have completed service. When the multiprogramming level is greater than one, i.e., there are several jobs that are being processed simultaneously in the system, synchronization must be acheived only between parallel tasks that belong to the same job. Therefore we must find a way to identify such tasks. This can be done by using colored tokens. In this case the initial marking is defined by the number of tokens in p1 and each token is given a distinct color. The state or marking of the colored GSPN is defined by the number and color of tokens in each place, i.e., the current marking M is given by,

$M = \{m1, m2, ..., mn\}$, and

$mi = \{mi1, mi2, ..mik\}$

where mij is the number of tokens of color j in place p_i. In this case a transition is enabled if there exists a token of the same color in each one of its input places. And it fires by removing these tokens and depositing a token of the same color in each one of its output places. For example transition t16 is enabled when there is a token of the same color in places 15 anf 16, and its firing resembles the synchronization of two parallel tasks that belong to the same job. Tokens in place 13, however, are not colored since they represent available resources at SC2. Therefore, transition t13 is enabled when there is a colored token in place 7, and a token in 13, and fires by taking these tokens and depositing a token in 10 with the same color as the

enabling token in 7. And transition t4 fires by taking a token from place 10 and depositing one with the same color in 17, and a noncolored one in 13.

It can be easily seen that the GSPN model is adequate for explicitly representing all key activities in a parallel processing system. However, for large scale systems with large number of service centers, and a workload with a large number of different types of tasks, the model becomes quite complex. The complexity of the model arises not only from the explicit representation of the various activities by places and transitions, but also due to the fact that the number of states (markings) will be prohibitively large. Therefore, the analysis of the model will be very complex. In the next section, an approximate hierarchical model, which is adequate for large scale systems, is presented.

4.4 An Approximate Hierarchical Model

In this section a hierachical model will be discussed. It employs both QNs and GSPNs. QNs are used to represent queuing and contention for multiple resources. And GSPNs are used to represent concurrent activities.

The model is based on multiple time scale decomposition. The time behaviour of the system can be divided in two time scales, a fast time scale, and a slow time scale. The fast time scale comprises activities describing the contention or execution of tasks at a certain resource in the system. And the slow time scale comprises activities describing the execution of tasks in

the system as a whole. The motivation behind the above assumption comes from the fact that, tasks in a system need several CPU-IO processing cycles before they terminate. Therefore, the soujorn time of a task in the system is much larger than the time needed by that task to execute at one of the resources during a cycle.

4.4.1 Model Discription

The model consists of two hierachical levels. At the lower level, the parallel processing system under consideration is represented by a multi-chain closed queuing network (QN). The system consists of a finite number K of servers including processors and IO devices, a particular pseudoserver labled 0, and a finite number of jobs N (the multiprogramming level). A job consists of several synchronous and asynchronous tasks. The pseudoserver, server 0, is a node in the network defined to accomodate changes in the number of tasks being processed in the system.

Let L denote the total number of the different types (statistically non-identical) tasks of the N jobs. The model assumes the availability of the routing behaviour as well as the service time distributions of the different types of tasks. Let P_{ijk} denote the probability that a type i task goes to server k after visiting sever j, and let S_{ij} denote the mean service time of type i task at server j. In order to make the model computationally feasible, we assume that the QN have a product form solution for a fixed number of the different types of tasks. Therefore, the queuing disciplines at each of the K servers are restricted to either, FCFS, processor sharing (PS), infinite server (IS), or last-come-first-served preemptive resume

(LCFSPR). The service times may have arbitrary distributions, except at FCFS servers in which case the service times of all tasks are exponentially distributed with a common mean, i.e., S_{ij} is independent of i if server j is a FCFS queue. These are the standard assumptions for a queuing network to have a product form solution [CHA77]. The QN can then be analyzed by the fast and simple Mean Value Analysis (MVA) algorithm of Reiser and Lavenberg [REI80]. However, an approximate analysis of some non-product form QN's, for example when a FCFS queue have non-exponential service time distribution, can be obtained using the extended MVA algorithm developed by Bard [BAR79].

At the higher level of the hierarchy, the behaviour or structure of jobs is modeled by means of a GSPN. In the GSPN, tokens represent tasks, places define the type and state of tasks in the QN (e.g., a task can be active, i.e., being processed, or is completed), and transitions resemble the activities of spawning, synchronization, or execution of tasks. An enabled timed transition resembles that a task is being processed in the system, and it fires when the task is completed. Therefore, the state (marking) of the GSPN defines the number of the different types of tasks competing for resources in the QN.

The QN is solved for each state M of the GSPN with different number of active tasks to determine the firing rates of enabled transitions. The rate of transition t_i ($r_i(M)$) is the throughput of type i tasks at node 0 of the QN, at state M. Here we also assume that the times between task arrivals at node 0 (task completions) are exponentially distributed. The correctness of

this assumption was proved by Courtios and Kleison [COU77,KEIL78]. The local performance parameters such as the steady state utilization $u_j(M)$, and mean queue length $l_j(M)$ of server j is also evaluated at each state M.

The GSPN is then solved for the steady state probability distribution. The global performance parameters are obtained from local ones as follows:

$R_i = \sum_{M \in S} r_i(M) \cdot P(M)$, $i=0,\ldots,L$

$U_j = \sum_{M \in S} u_j(M) \cdot P(M)$, $L_j = \sum_{M \in S} l_j(M) \cdot P(M)$, $j=1,\ldots,K$

where, P(M) (M S), is the steady state probability distribution ,and S is the state space or reachability set of the GSPN. Also R_i is the throughput of type i task at node 0, U_j and L_j are the global utilization and mean queue length of server j respectively.

To further illustrate the above model, let us consider the following two examples:

Example 4.3: Asynchronous tasks.

Let us consider the case where there are N statistically identical (s.i.) jobs in the system. Each job consists of a primary task (type 0), and one or more s.i. asynchronous (type 1) tasks initiated one at a time by the primary task, and execute concurrently with it and terminate independently. Also consider the central server QN model shown in figure 4.4. The model consists of a processor queue, and an IO queue. The processor queue is assumed to be a single server queue with a PS queuing discipline, and the IO queue is single server queues with FCFS queuing discipline. Figure 4.5 shows the SPN model of jobs behaviour (with N=2). The state of the SPN is M=(N,k), where N is

the number of primary tasks (tokens in p_1), and k is the number of spawned secondary tasks (tokens in p_2). The firing of transition $t_\emptyset$ resembles the initiation of a secondary task, where a token is placed in p_2. The firing of t_1 resembles the termination of a secondary task, where a token is taken out of p_2. The firing rates of these transitions $r_\emptyset(N,k)$ and $r_1(N,k)$ are the throughputs of N type $\emptyset$ and k type 1 tasks at node $\emptyset$ of the QN, respectively. The corresponding M.C of the SPN is shown in figure 4.6. This M.C obeys local balance, i.e., at steady state, the balance equations are,

$$r_\emptyset(2,k-1).P(k-1) = r_1(2,k).P(k) \qquad k>\emptyset,$$

and with $\sum_{k=\emptyset}^{\infty} P(k)=1$, the steady state probability distribution is then,

$$P(k) = P(\emptyset).\ \prod_{i=\emptyset}^{k-1} r_\emptyset(2,i)/r_1(2,i+1) \qquad k>\emptyset, \text{ and}$$

$$P(\emptyset) = 1 \ /\ (\ 1+ \sum_{k=1}^{\infty} \prod_{i=\emptyset}^{k-1} r_\emptyset(2,i)/r_1(2,i+1)\ )$$

The global performance parameters are evaluated as follows:

$$R_i = \sum_{k=\emptyset}^{\infty} r_i(2,k)\ .\ P(k) \qquad i=\emptyset,1,$$

$$U_j = \sum_{k=\emptyset}^{\infty} u_j(2,k)\ .\ P(k), \text{ and } Q_j = \sum_{k=\emptyset}^{\infty} q_j(2,k).P(k) \qquad , j=1,2,\ldots,n+1$$

The existence of the above solution can be easily proved since $r_\emptyset(2,k)/r_1(2,k) < 1$ for all $k>2$. Therefore the term $\prod_{i=\emptyset}^{k-1} r_\emptyset(2,i)/r_1(2,i+1)$ rapidly tends to zero as k increases, and the above infinite sum can be accurately approximated by taking a finite number of terms.

Example 4.4: Synchronous tasks.

Let us consider the case where a primary task subdivides into exactly two secondary (type 1, and type 2) synchronous concurrent tasks. When a secondary task completes, it must wait until its

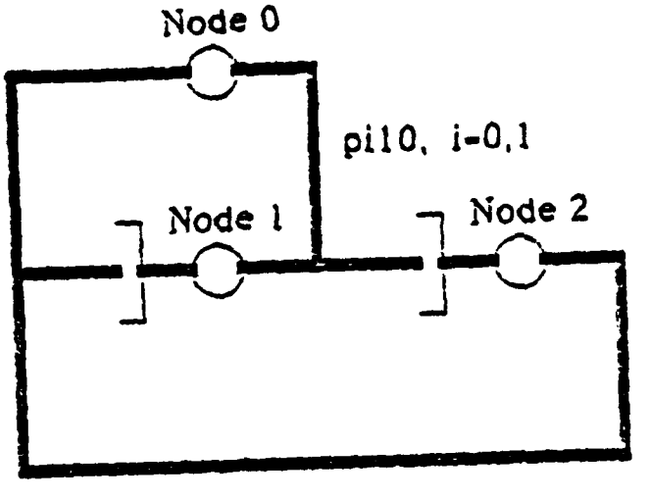


Figure 4.4

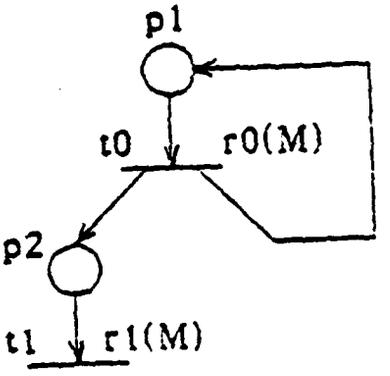


Figure 4.5

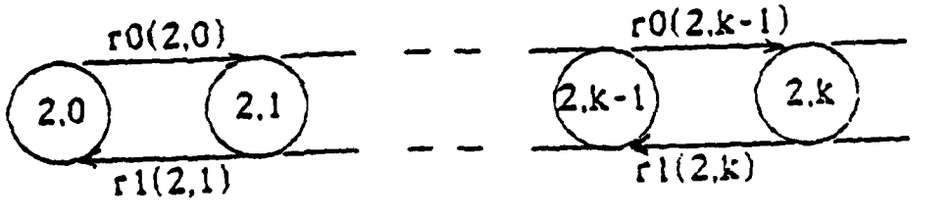


Figure 4.6

sibling has completed, at which time the primary task becomes active again and the process is repeated. The GSPN shown in figure 4.7 resembles this behaviour (N=1). The number of tokens in p_2 and p_3 are equal to the number of active type 1 and type 2 tasks respectively. The number of tokens in p_4 and p_5 are the number of completed tasks that are waiting for their siblings. The reachability set of the GSPN is shown in table 4.1. And the corresponding MC is shown in figure 4.8. As mentioned in the previous section, if N > 1, the tokens are colored to acheive proper synchronization between the secondary tasks. Therefore, each token defined initially in p_1 should have a distinct color.

Now consider the simple three chain QN model shown in figure 4.9 Both the processor and IO queues are two server queues. Since with N=1 there will be a most two tasks in the system at any given time, the queues are effectively IS (no contention).

Let the mean service time of the above tasks be,

$S_{i1} = S_{i2} = 1$ sec. , i=0,1,2. Also the routing probabilities $P_{i10} = p$,i=0,1,2. Since the mean service times for a task at both queues are equal, the probability of finding at least one task of a certain type at any one of the queues is 1/2. Therefore the throughputs of tasks at node 0 are,

$r_0(1) = 1/2\ 1/S_{01}\ P_{010} = p/2 = r_i(j)$, i=1,2 , and j=2,3,4.

The rate transition matrix of the above M.C. is

$$A = p \begin{bmatrix} -1/2 & 1/2 & 0 & 0 \\ 0 & -1 & 1/2 & 1/2 \\ 1/2 & 0 & -1/2 & 0 \\ 1/2 & 0 & 0 & -1/2 \end{bmatrix}$$

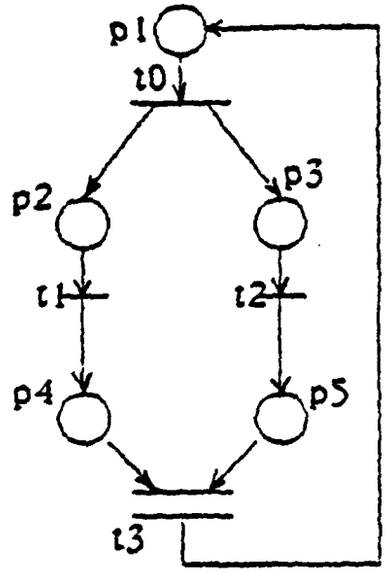


Figure 4.7

	p1	p2	p3	p4	p5
1	1	0	0	0	0
2	0	1	1	0	0
3	0	0	1	1	0
4	0	1	0	0	1
5	0	0	0	1	1

Table 4.1

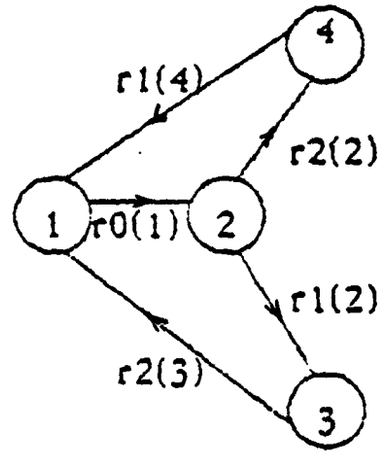


Figure 4.8

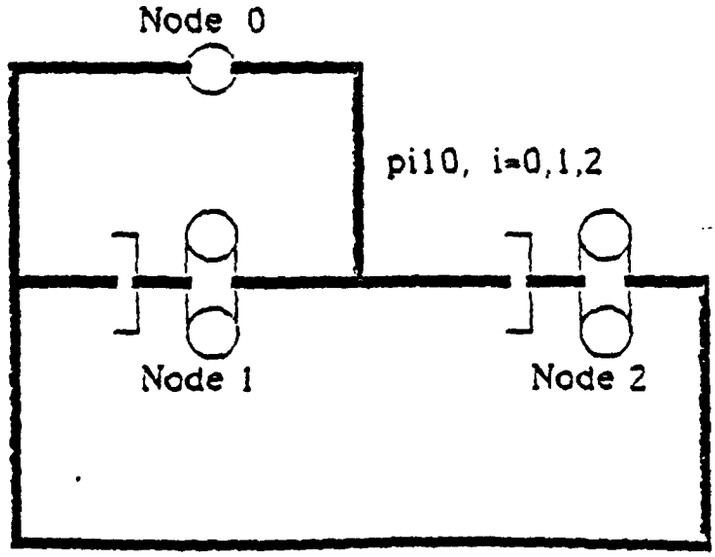


Figure 4.9

Using the above matrix, the steady state probability distribution of the GSPN can be evaluated, and the global performance parameters can be obtained as indicated earlier.

4.4.2. Theoretical Verification

The development of the above hierarchical model is based on the concept hierarchical aggregation of continuous-time discrete-state Markov processes. The theory presented here was developed in two recent papers by Coderch et al [COD83a,b], which generalizes the earlier work on decomposition of Markov chains proposed by Courtios [COU77].

Let $\{X^p(t),\ t>=0\}$ be a finite state markov process (FSMP) with rare transitions. The transition probability matrix of this process is $P^p(t) = \exp\{A(p)t\}$, where

$$A(p) = \sum_{i=0}^{\infty} p^i A_{0i} \qquad (3.1)$$

is the matrix of transition rates, and $p\ [0,p_o]$ is a small parameter modeling rare transitions in $X^p(t)$. This process can be considered to be a perturbed version of the FSMP $X^o(t)$, where these rare transitions are neglected ($p=0$).

Definition: $X^p(t)$ is said to be regularly perturbed if

$$\lim_{p\to 0} \sup_{t>=0} \left\| P^p(t) - P^o(t) \right\| = 0$$

otherwise, the perturbation is singular. In which case rank $A(p)$ > rank A_{00}, or equivalently, the number of ergodic classes at of $X^p(t)$ is less than that of $X^o(t)$.

If $X^p(t)$ is a singularly perturbed FSMP, then

i) the limits

$$\lim_{p\to 0} P^p(t/p^k) = P_k(t)\ ,\quad k=1,2,\ldots,m.$$

are all well defined and determine a finite sequence of (in

general stochastically discontinuous) FSMP's $X_k(t)$, k=1,..,m with transition probability matrix $P_k(t)$ (refer to section 2 of the next chapter for the detailed definition of stochastically discontinuous FSMP's);

ii) the limit processes $X_k(t)$ can be aggregated to produce a hierarchy of simplified, approximate models of $X^p(t)$, each of which is a FSMP valid at a certain time scale t/p^k describing changes in $X^p(t)$ at a distinct level of detail; and

iii) the collection of the aggregated models $X_k'(t)$, k=1,...,m can then be combined to construct an asymptotic approximation of $X^p(t)$ uniformally valid for $t>=0$.

The above can be expressed by the following theorem:

Theorem 1: Let $X^p(t)$ be a singularly perturbed stochastically continuous FSMP taking values in $E_0=\{1,2,..,n_0\}$, with transition probability matrix $P(t) = \exp\{A(p)t\}$, and infinitismal generator $A(p)$ of the form (3.1), then

i) let A_k and Z_k be respectively the infinitismal generator and the ergodic projection at zero of some FSMP $X_k(t)$ taking values in E_0,

$$\lim_{p\to 0} \sup_{t \in [d,T]} \left\| P(t/p^k) - Z_k \exp\{A_k t\} \right\| = 0$$

for all $d>0$, and $T<\infty$ and k=1,..,m (T can be taken equal to ∞ for k=m). Furthermore, let $Z_k = V_k.U_k$ be the canonical product decomposition of Z_k (see proposition 2.1 of the next chapter) . Then,

ii)

$$P(t) = \exp\{A(p)t\} = \exp\{A_{00}t\} + \sum_{k=1}^{m} (V_k \exp\{A_k p^k t\} U_k - Z_k) + o(1)$$

uniformally for $t>=0$, where $A_k' = U_k A_k V_k$ is the generator of a

FSMP $X_k'(t)$ taking values in $E_k=\{1,..,n_k\}$ and

$$n_k = n_\emptyset - \sum_{i=\emptyset}^{m} \text{rank } A_i$$

Proof: [COD83b]

The matrices A_k' and Z_k are evaluated for k=∅,1,2, as follows:

$$A_\emptyset' = A_\emptyset = A_{\emptyset\emptyset} \ ,$$

$$A_1' = U_1 A_1 V_1 = U_1 A_{\emptyset 1} V_1 \ ,$$

$$A_2' = U_2 A_2 V_2 = U_2 (A_{\emptyset 2} - A_{\emptyset 1} A^{\#} A_{\emptyset 1}) V_2 \ ,$$

where $A^{\#} = (A_{\emptyset\emptyset} + Z_1)^{-1} - Z_1$;

and $Z_1 = \lim_{t\to\infty} \exp\{A_\emptyset t\}$, or is the solution of

$A_\emptyset Z_1 = \emptyset$, with $Z_1 1^+ = 1^+$, where $1^{+T} = [1\ 1\ .\ .\ .1]$

$Z_2 = \lim_{t\to\infty} Z_1 \exp\{A_1 t\} = V_1 (\lim_{t\to\infty} \exp\{A_1 t\}) U_1$,

or can be evaluated from

$A_1' Z_1' = \emptyset$, with $Z_1' 1^+ = 1^+$ and then

$Z_2 = V_1 Z_1' U_1$

For the expressions for higher order models refer to [COD83b]. Also Delebeque [DEL83] developed a recursive algorithm to compute the above aggregated models A_k'.

Example 4.5:

To illustrate the above theory, consider the simple FSMP $X^p(t)$, the state rate transition diagram of which is shown in figure 4.10. With the transition rate p between states 2 and 3 is much less than one, the process will spend a random amount of time switching between states 1 and 2 and eventually it will get trapped in 3. It is clear that we can identify phenomena occuring at two time scales. At the "fast" time scale only transitions between 1 and 2 occur and $X^o(t)$ is a good model for that. At the "slow" time scale (t/p) a sample function of the process in the

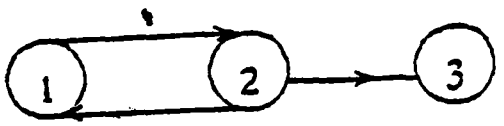


Figure 4.10

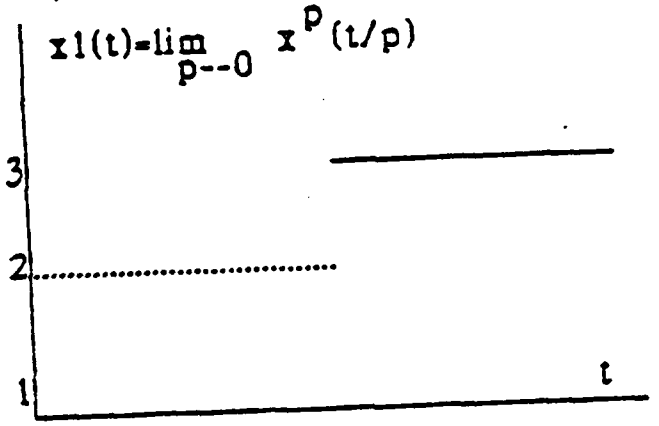


Figure 4.11

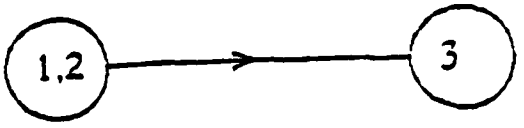


Figure 4.12

limit $p \to \emptyset$ is shown in figure 4.11, which is denoted by $X_1(t)$. It is clear that this function has an infinite number of discontinuities on a finite time interval. The distribution of the random variable describing the length of this interval converges to the exponential distribution [KEI78]. This process can then be approximately aggregated (figure 4.12). The rate transition matrix of the aggregated model is obtained as follows:

Let the rate transition matrix of $X^p(t)$ be expressed as,

$$A(p) = A_{\emptyset\emptyset} + p\, A_{\emptyset 1}, \quad A_{\emptyset\emptyset} = \begin{bmatrix} -1 & 1 & \emptyset \\ 1 & -1 & \emptyset \\ \emptyset & \emptyset & \emptyset \end{bmatrix}, \text{ and}$$

$$A_{\emptyset 1} = \begin{bmatrix} \emptyset & \emptyset & \emptyset \\ \emptyset & -1 & 1 \\ \emptyset & \emptyset & \emptyset \end{bmatrix}, \text{ then solving for the steady state}$$

probability matrix Z_1 of the process $X^o(t)$,

$A_{\emptyset\emptyset}\, Z_1 = \emptyset$ with $Z_1\, 1^+ = 1^+$, we have

$$Z_1 = \begin{bmatrix} 1/2 & 1/2 & \emptyset \\ 1/2 & 1/2 & \emptyset \\ \emptyset & \emptyset & 1 \end{bmatrix} = V_1\, U_1 = \begin{bmatrix} 1 & \emptyset \\ 1 & \emptyset \\ \emptyset & 1 \end{bmatrix} \begin{bmatrix} 1/2 & 1/2 & \emptyset \\ \emptyset & \emptyset & 1 \end{bmatrix}$$

then using the above expressions,

$$A_1' = U_1\, A_{\emptyset 1}\, V_1 = \begin{bmatrix} -1/2 & 1/2 \\ \emptyset & \emptyset \end{bmatrix}$$

Also notice that rank $A_{\emptyset\emptyset}$ + rank A_1' = rank $A(p)$, and therefore by theorm 3.1,

$$\exp\{A(p)t\} = \exp\{A_{\emptyset\emptyset}t\} + V_1 \exp\{A_1\, p\, t\}\, U_1 - Z_1 + o(1)$$

for small t $\exp\{A_1\, p\, t\} = I$, $\exp\{A(p)t\} = \exp\{A_{\emptyset\emptyset}t\}$, and for large t $\exp\{A_{\emptyset\emptyset}t\} = Z_1$ $\exp\{A(p)t\} = V_1 \exp\{A_1 pt\}\, U_1$

To show the application of the above theory to the hierarchical model described in previous section, let us consider example 4.4 of that section. And let us analyze the exact solution of the QN in figure 4.9 under the specified job

behaviour using the Markov chain model. And assuming that the service times distributions for all of the three types of tasks in the network are exponential. The M.C. model of the QN in figure 4.9 is shown in figure 4.13. The state is characterized by a 2X3 matrix with raws representing queues and columns representing the task types.

For example at state 1, a type Ø task is being serviced at queue 1 (the processors queue), while at state 2 it is being serviced at the IO queue. The transition rate from state 1 to state 2 is equal to $1/S_{Ø1} \cdot (1-P_{Ø1Ø}) = 1 - p$, where $S_{Ø1}=1$ and $P_{Ø1Ø} = p$

The limit as p-- Ø can be interpreted as that the average number of cycles that a task makes through the processor queue and the IO queue tends to infinity. If the tasks are large, i.e., they need several cycles of CPU IO procesing (p<<1) before they terminate (branch to node Ø), the above M.C. can be analyzed using the theory described above. Let the rate transition matrix of this of this M.C. be expressed as

$$A(p) = A_{ØØ} + p\,A_{Ø1} \text{ , where } A_{ØØ} = \begin{bmatrix} A" & Ø & Ø & Ø \\ Ø & B" & Ø & Ø \\ Ø & Ø & C" & Ø \\ Ø & Ø & Ø & D" \end{bmatrix}$$

$$A" = C" = D" = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}, \quad B" = \begin{bmatrix} -2 & 1 & 1 & Ø \\ 1 & -2 & Ø & 1 \\ 1 & Ø & -2 & 1 \\ Ø & 1 & 1 & -2 \end{bmatrix} \text{ , and}$$

$$A_{Ø1} = \begin{bmatrix} Ø & -1 & 1 & Ø & Ø & Ø & Ø & Ø & Ø & Ø \\ Ø & Ø & Ø & Ø & Ø & Ø & Ø & Ø & Ø & Ø \\ Ø & Ø & Ø & -1 & -1 & Ø & 1 & Ø & 1 & Ø \\ Ø & Ø & Ø & Ø & Ø & -1 & Ø & Ø & Ø & 1 \\ Ø & Ø & Ø & Ø & Ø & -1 & Ø & 1 & Ø & Ø \\ Ø & Ø & Ø & Ø & Ø & Ø & Ø & Ø & Ø & Ø \\ 1 & Ø & Ø & Ø & Ø & Ø & Ø & -1 & Ø & Ø \\ Ø & Ø & Ø & Ø & Ø & Ø & Ø & Ø & Ø & Ø \\ 1 & Ø & Ø & Ø & Ø & Ø & Ø & Ø & Ø & -1 \\ Ø & Ø & Ø & Ø & Ø & Ø & Ø & Ø & Ø & Ø \end{bmatrix}$$

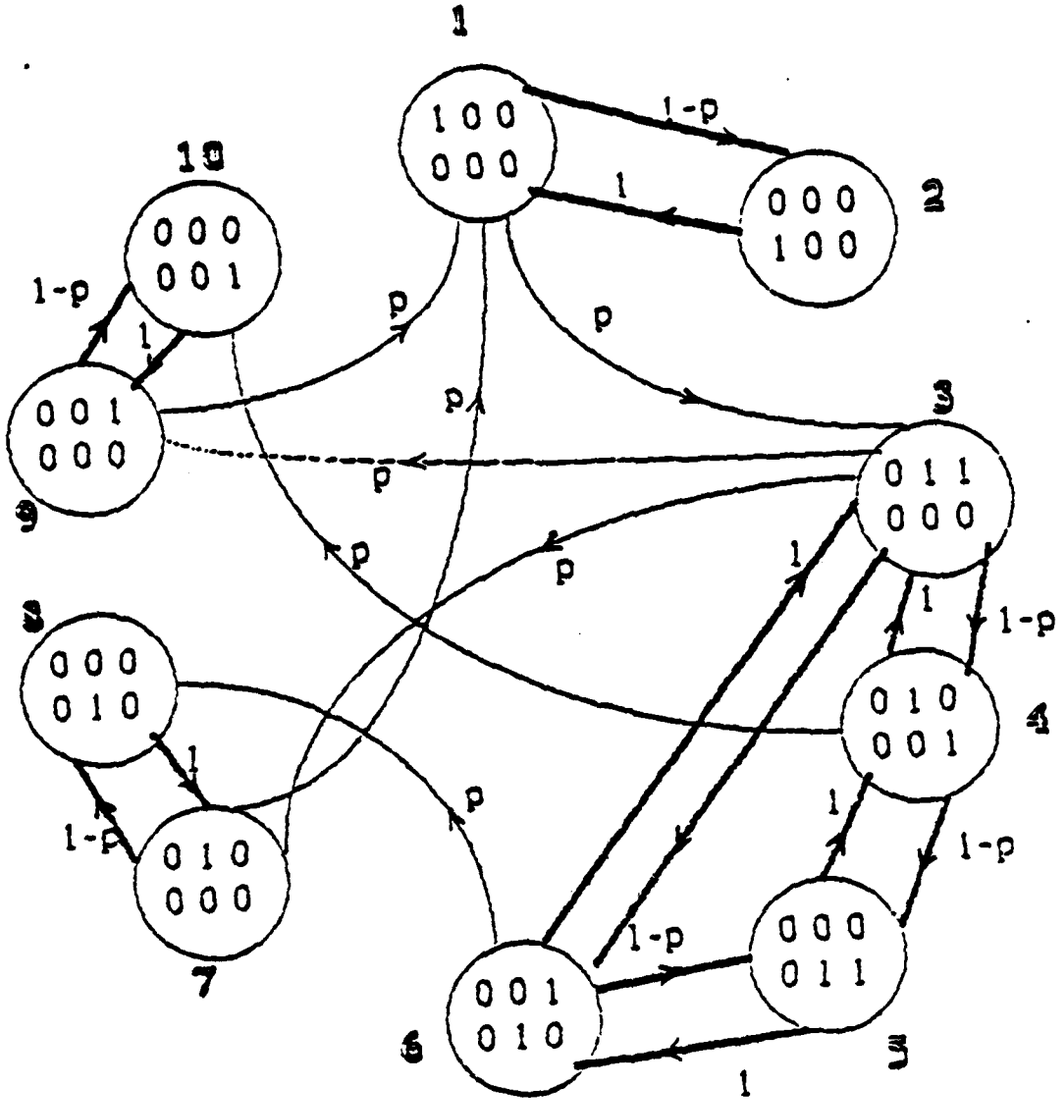


Figure 4.13

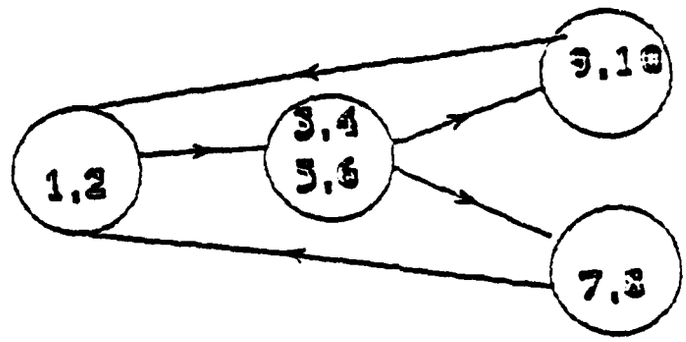


Figure 4.14

It can be easily seen that the M.C. is decomposed into four ergodic classes at zero (i.e., at p=0), with transitions represented in the figure by solid lines as follows:

E_1={1,2} , E_2={3,4,5,6} , E_3={7,8} , and E_4={9,10}.

These classes partition the M.C. into four chains with rate transition matrices given by A", B", C", and D" respectively. In chain 1 the type 0 task is being processed in the network in either the processor queue or the IO queue, in chain 2 both type 1 and type 2 tasks are being processed, in chain 3 the type 2 task only is being processed (type 1 has terminated first), and in chain 4 the type 1 task is being processed (type 2 has terminated first). The steady state probability matrix Z_1 can be evaluated by solving each one of these chains separatly for the steady state probability matrices Z_i" , i=1,2,3,4, which are

$$Z_i" = \begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix} , \; i=1,3,4 \; , \text{ and } Z_2" = [z_{ij}], \; z_{ij} = 1/4 \; , \; i,j=1,2,3,4$$

$$\text{Then, } Z_1 = \begin{bmatrix} Z_1" & & & \\ & Z_2" & & \\ & & Z_3" & \\ & & & Z_4" \end{bmatrix} = V_1 \cdot U_1$$

$$\text{and } A_1' = U_1 \, A_{01} \, V_1 = \begin{bmatrix} -1/2 & 1/2 & 0 & 0 \\ 0 & -1 & 1/2 & 1/2 \\ 1/2 & 0 & -1/2 & 0 \\ 1/2 & 0 & 0 & -1/2 \end{bmatrix} , \text{ which is}$$

the rate transition matrix of the aggregated M.C. shown in figure 4.14. This M.C. is the same as the one in figure 4.8, which corresponds to the GSPN model in figure 4.7. Therefore, the GSPN

models the system from a higher level of detail defining the number of the different types of active tasks in the system. And the corresponding M.C. model of the GSPN is the aggregated model that describes the behaviour of the process at the slow time scale (the time required to process a whole task). While at the fast time scale (the time required to process a portion of the task at one of the servers, i.e., during one cycle in the system) the QN models the behaviour of the process at a lower level of detail, i.e., the types of servers required and the mean service time at each server per visit.

The above example demonstrates that the approximate hierarchical model proposed in the previous section is based on the approximate hierarchical aggregation of the exact M.C. model. And the basic assumption is that jobs consists of large tasks ,i.e., tasks that require multiple accesses to many different system resources before they terminate.

4.4.3 Validation Examples

In this section, the accuracy of the above model is validated by comparison to discrete event simulation. Two examples of systems with asynchronous tasks and synchronous-asynchronous concurrent tasks are considered.

In the first example, the system described in example 4.3 ,with jobs consisting of asynchronous tasks, is considered. This system was simulated by a simulation program writen in SIMSCRIPT II.5. The analytical results, obtained from the hierarchical model as described in example 4.3, were found to be very accurate compared to the simulation results for a variety of models. Table

4.2 shows the parameter settings for 10 different models (figure 4.4). These parameters were chosen such that the IO server (server 2) or the CPU server (server 1), or both are heavily utiltized. This corresponds to cases 1-4, cases 6-9, and cases 5 and 10, respectively. The multiprogramming level N is equal to 5 in all cases. For simplicity, The service time distributions of both tasks at the IO server are assumed to be exponential with the same mean S_2 so that the underlying QN will have a product form solution. The following parameters are common to all cases,

S_{11} = 0.00001 , and S_2 = 0.0002

Model number	p010	p110	S01
1	0.1	0.1	0.0001
2	0.3	0.1	0.0001
3	0.5	0.1	0.0001
4	0.1	0.3	0.0001
5	0.1	0.3	0.0001
6	0.1	0.1	0.001
7	0.1	0.3	0.001
8	0.1	0.5	0.001
9	0.3	0.1	0.001
10	0.5	0.1	0.001

Table 4.2 Parameter settings for central server models.

Table 4.3 shows the results obtained for the throughputs of tasks at the CPU, and servers utilization. The simulation results are shown between parenthesis followed by the relative percentage error between the analytical and simulation results.

MODEL NO.	THROUGHPUTS AT CPU OF TASK TYPE 0	1	UTILIZATION CPU	IO
1	2777 (2783),.21	2777 (2780),.1	.31 (.31)	.999 (1.00)
2	1470 (1475),.33	4417 (4413),.04	.19 (.21),1.8	1.0 (1.0)
3	1003 (1007),.29	4998 (5007),.17	.15 (.17),11	1.0 (1.0)
4	4379 (4399),.45	1459 (1460),.06	.45 (.44),2.2	.99 (.99)
5	4926 (4951),.5	985 (988),.35	.49 (.50),2	.98 (.99), 1
6	989.7 (998.5),.88	989.7 (981.5),.83	1.0 (.99),1	.36 (.36)
7	996 (1003),.7	332.2 (332.1),.03	.99 (.99)	.23 (.23)
8	997 (1002),.5	199.6 (200.1),.2	1.0 (.97),3	.20 (.20)
9	969.7 (981.7),1.2	2909 (2912),.1	1.0 (.99),1	.66 (.66)
10	935.9 (951.9),1.6	4679 (4734),1.1	.98 (.98)	.94 (.94)

Table 4.3. Comparison between analytical and simulation results.

Notice that the analytical results are still very accurate even when one of the servers is saturated (100% utilization). This was not the case in the model developed in [HED 82] which is mainly suitable for balanced systems.

Figure 4.15 shows the throughput of primary tasks as a function of the mutliprogramming level N from the analytical and

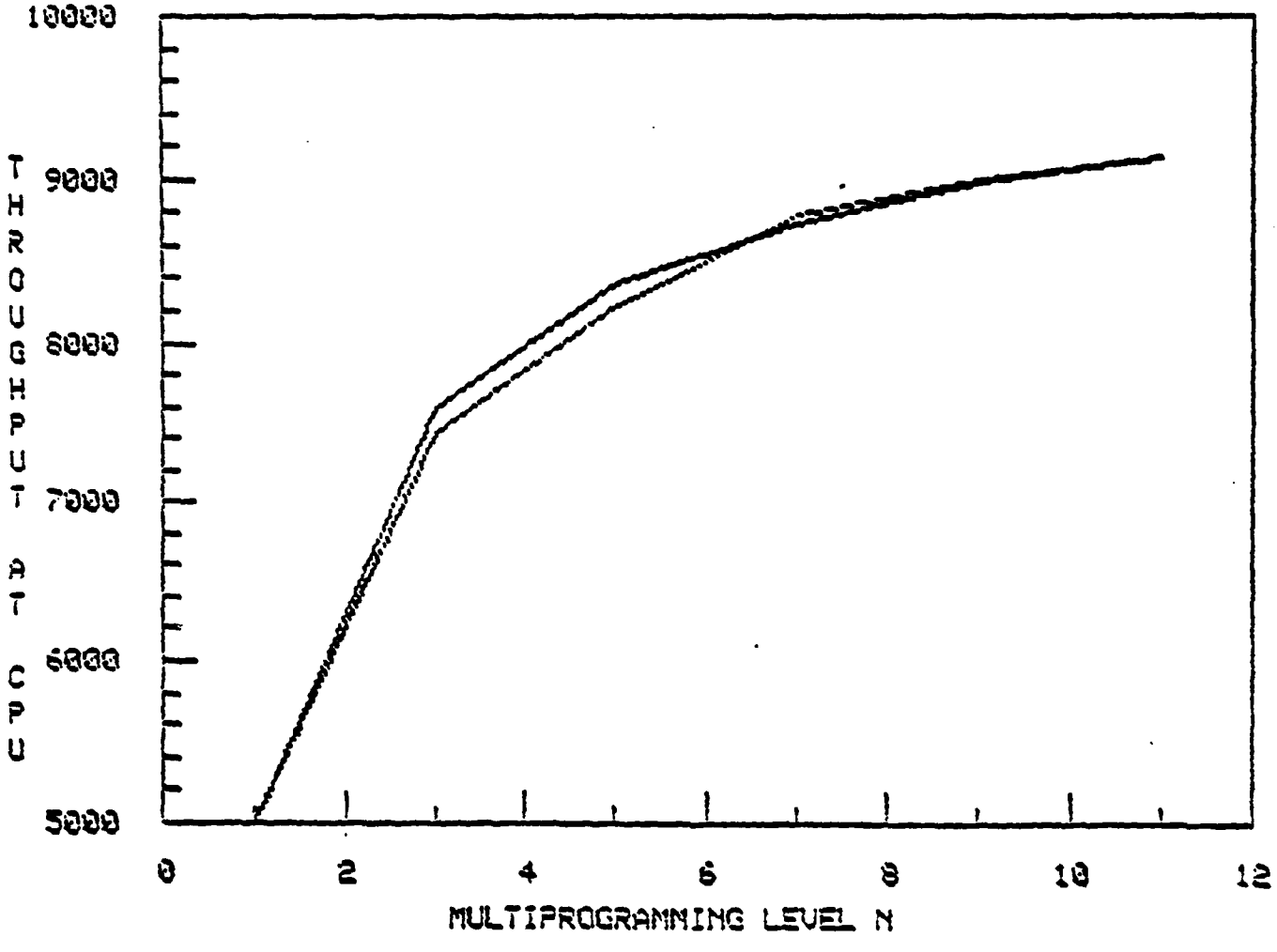

10000
9000
8000
7000
6000
5000
THROUGHPUT AT CPU
0
2
4
6
8
10
12
MULTIPROGRAMMING LEVEL N

Figure 4.15

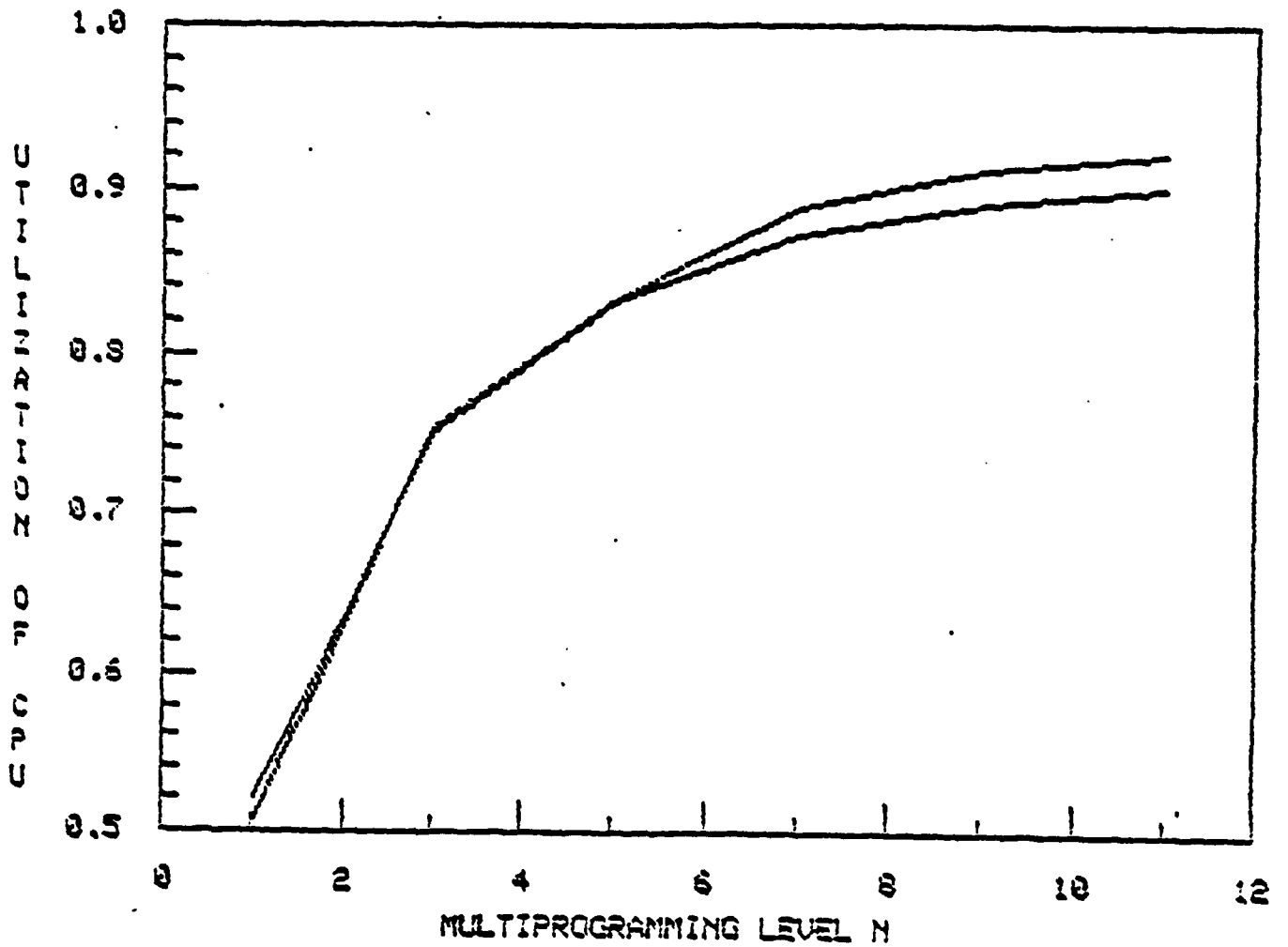

1.0
0.9
0.8
0.7
0.6
0.5
CPU UTILIZATION
0
2
4
6
8
10
12
MULTIPROGRAMMING LEVEL N

Figure 4.16

simulation models. And figure 4.16 shows the the CPU utilization as a function of N. The parameters for these figures are as follows,

S_{01} = .0001, S_{11} = .00001, S_2 = .0001, P_{010} = .05, and P_{110} = .4

Example 4.7: Consider the QN in figure 4.17 with a CPU queue at node 1, and an IO queue at node 2. Let us assume again for simplicity that the CPU queue is a single server queue with processor sharing queueing discipline, and exponentially distributed service times with mean S_{i1} for type i tasks. The IO queue is also a single server queue with FCFS queueing discipline, and exponentially distributed service times with common mean for all types of tasks, i.e., S_{i2} is independent of i. This QN will have a product form solution for a specific number of tasks.

Jobs behaviour is modeled by the GSPN in figure 4.18 , where a primary task may with probability P(2) subdivide into two synchronous (type 1 and type 2) tasks, or with probability (1-P(2)) spawn an asynchronous task (type 3), which executes concurrently with it and terminates independently. The markovian model of this GSPN is complicated to evaluate. But using the concept of hierarchical decomposition on the GSPN the above system can be easily solved. If we assume that the initiation of synchronous tasks is more frequent than asynchronous ones (P(2) > 0.5), and that asynchronous tasks take longer time to execute than synchronous ones. Then, while a certain number of asynchronous tasks are executing in the system, several synchronous ones will start and terminate for some time enough to

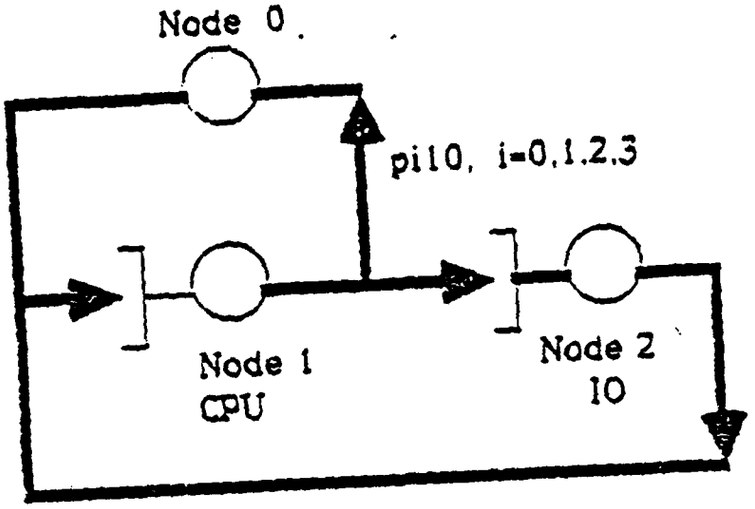


Figure 4.17

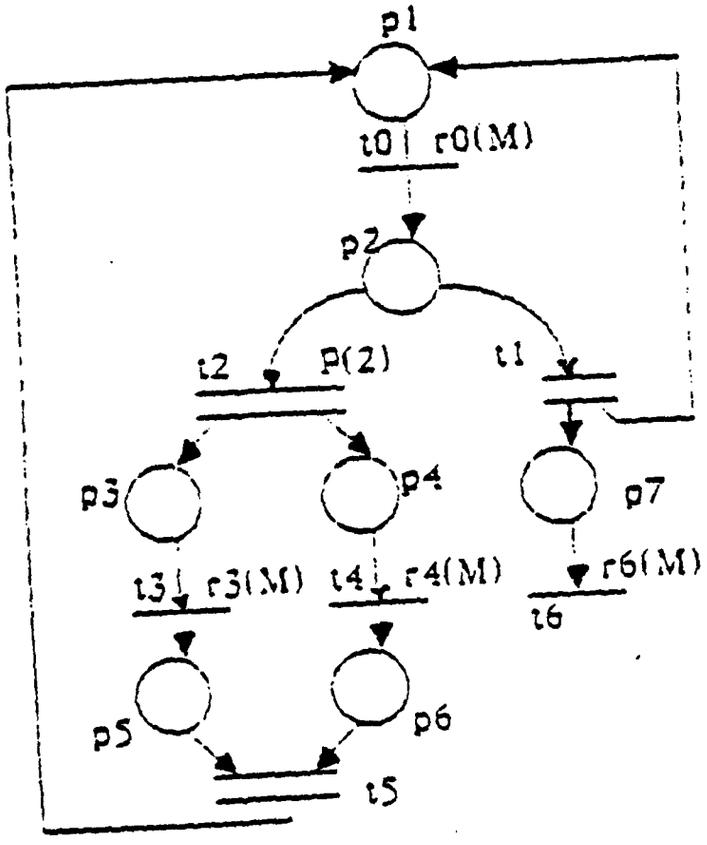


Figure 4.18

let the subnetwork that models their activities reache local equilibrium. The GSPN can then be decomposed to the GSPN (N1) in figure 4.7 of example 4.4 at the lower level, and the SPN (N2) in figure 4.5 of example 4.3 at the higher level of the hierarchy.

In figure 4.19, let rØ1 and rØ2 be the rates of transition tØ for N1 and N2 respectively. Then, at each state k of N2 which is defined by the number of tokens in p7 (i.e, for a certain number of asynchronous tasks in the system), N1 is to be solved for the local steady state probability distribution $P_1(M',k)$, $M' \quad S_1$, where S_1 is the reachability set of N1 when there are N tokens initially in p1 (figure 4.19 (a)). And $r\emptyset1(M',k) = P(2).r\emptyset(M',k)$. Also the local performance parameters such as throughputs, utilizations, and mean queue lengths are to be evaluated. Then N2 can be solved as mentioned in example 4.3, with

$$r_{\emptyset 2}(k) = (1-P(2)) \sum_{M' \in S1} r_{\emptyset}(M',k) \; P_1(M',k),$$

and $$r_6(k) = \sum_{M' \in S1} r_6(M',k) \; P_1(M',k).$$

Where $r_{\emptyset}(M',k)$ and $r_6(M',k)$ are the throughputs of type Ø and type 3 tasks respectively at node Ø of the QN at state M=(M',k). And the global performance parameters can be evaluated from local ones as mentioned before.

The above was implemented for a set of ten central server models. Table 4.4 shows the different parameters of the models. This set contains cases with only moderate utilizations at the devices as well as heavily CPU and / or IO bound cases. The following parameters are common to all cases;

N =2, $p_{11\emptyset} = 1 - p_{112} = \emptyset.9$, $p_{21\emptyset} = 1 - p_{212} = \emptyset.9$,

$p_{31\emptyset} = 1 - p_{312} = \emptyset.1$, $s_{\emptyset 1} = \emptyset.\emptyset1$, and $s_{31} = \emptyset.\emptyset\emptyset\emptyset3$

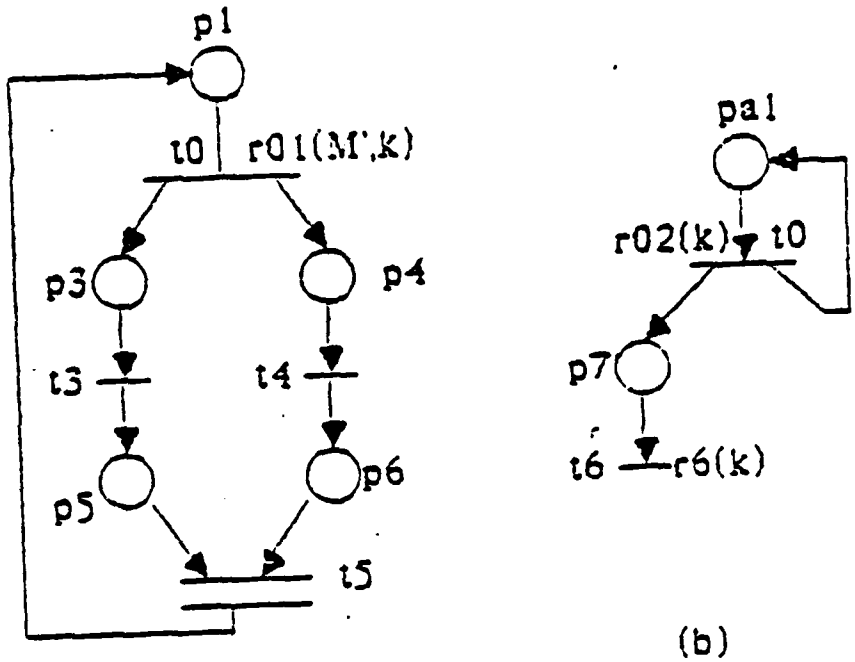


Figure 4.19

Model number	P(2)	pØ1Ø	S11	S21	S2
1	Ø.9	Ø.5	Ø.5	Ø.5	Ø.1
2	Ø.7	Ø.5	Ø.5	Ø.5	Ø.1
3	Ø.5	Ø.5	Ø.5	Ø.5	Ø.1
4	Ø.9	Ø.1	1	Ø.5	Ø.Ø4
5	Ø.5	Ø.1	1	Ø.5	Ø.Ø4
6	Ø.3	Ø.1	1	Ø.5	Ø.Ø4
7	Ø.9	Ø.1	Ø.Ø1	Ø.Ø5	Ø.ØØ8
8	Ø.7	Ø.1	Ø.Ø1	Ø.Ø5	Ø.ØØ8
9	Ø.5	Ø.1	Ø.Ø1	Ø.Ø5	Ø.ØØ8
1Ø	Ø.3	Ø.1	Ø.Ø1	Ø.Ø5	Ø.ØØ8

Table 4.4 Paramerter settings for central server models

Notice that the synchronous tasks are CPU bound and the asynchronous tasks are IO bound tasks.

Each of the above models was simulated by a simulation program written in SIMSCRIPT II.5. And each simulation was run for several minutes on an IBM 3033.

Table 4.5 shows the percent relative errors (1ØØ percent times the absolute value of simulation estimate minus approximate analytical value divided by simulation estimate) for the performance parameters of each model.

MODEL NO.	THROUGHPUTS AT CPU OF TASK TYPE 0	1	2	3	UTILIZATION CPU	IO	MEAN QUEUE LENGTH AT CPU
1	0.1	0.3	0.0	2.2	0.5	0.0	4
2	0.6	1.6	1.6	0.5	1.9	0.1	0.5
3	14	14	15	15	15	16	20
4	2.2	1.8	1	0.1	2.2	3.8	7.2
5	4.6	4.3	4.3	6.3	5.2	2	0.1
6	15	17	17	17	19	15	10
7	5.1	5.2	4.9	6.9	4.7	5.3	6.1
8	3.1	1.4	2.1	2.1	2.7	2.8	2.5
9	0.0	1.1	0.4	3.8	0.6	1.3	2.5
10	4.8	6.3	6.3	6.3	5.3	5.6	10

Table 4.5. Percent relative errors

Large errors were found in models with high IO utilization (around 90%), such as models 3 and 6. Best results were found in the CPU bounded models, as expected. The above decomposition of GSPNs will be investigated further in Chapters 6 and 7.

CHAPTER 5

ANALYSIS OF THE GENERALIZED STOCHASTIC PETRI NETS BY STATE AGGREGATION

5.1 Introduction

In this chapter, the analysis of GSPNs will be considered. The existing methods of analysis will be briefly described with their advantages and limitations. A different and more general method of analysis will then be presented.

As mentioned before there are two types of transitions in GSPNs, immediate, and timed. Once enabled, immediate transitions fire in zero time, while timed ones fire in an exponentially distributed random time. Several transitions may be enabled by a marking. If the set of enabled transitions H comprises only timed transitions with rates r_i ($i \in H$), then the enabled timed transition t_i fires with probability

$$r_i / \sum_{k \in H} r_k \qquad (5.1.1)$$

If H comprises several timed transitions and one immediate transition, then this is the one that fires with probability one. If H comprises several immediate transitions, it is necessary to specify a probability distribution on the set of enabled immediate transitions according to which the firing transition is selected. The subset of H comprising all enabled immediate transitions together with the associated probability distribution is called a random switch, and the associated distribution is called the switching distribution.

Assuming that the reachability set S is finite, and firing rates of timed transitions do not depend on the time parameter

(however they may be marking or state dependent), Marsan et al [MAR84] have recognized that the time behaviour of a GSPN is equivalent to a stationary (homogenous), finite state, continuous time stochastic point process (SPP). And that a one to one correspondance exist between GSPN markings and the SPP states. The sample functions of the SPP may present "multiple discontinuities" due to the sequential firing of one or more immediate transitions. The process is observed to spend a non-negative amount of time in markings enabling timed transitions only, while it transits in zero time through markings enabling immediate transitions. It is called tangible a state (or a marking) of the former type and vanishing a state (or a marking) of the latter type.

Therefore, the state space of the GSPN is divided into, a set of tangible states, and a set of vanishing states. Furthermore, by assuming that the GSPN is irreducible, i.e., each element of the set of all possible markings S is reachable with a non-zero probability from any other state of the set (no marking, or a group of markings exists that absorbs the process), they proposed two solution methods for evaluating the steady state probability distributions of the GSPN.

The first method, which is a simple extension of the one proposed by molloy [MOL81], assumes that all immediate transitions are replaced by timed transitions characterized by very high firing rates propotional to an arbitrary value x. Under this assumption all states are tangible, and the GSPN reduces to a standard SPN, which can be analyzed by solving the corresponding M.C.. If an explicit solution expression for the

probability distribution of this SPN is obtained, the steady states probability distribution of the original GSPN can be obtained by taking the limit for x going to infinity of such solution. However, since most practical cases involve GSPNs with a large state space, an explicit expression of the solution in terms of x is usually not easy to obtain, and the practical approach that can be suggested of numerically solving the problem by assuming x to be very large and setting to zero those probabilities that appear exceptionally small, is confronted by numerical problems. Moreover, the above method not only requires useless computations of the probabilities of vanishing states, but it also increases the computational complexity by enlarging the size of the rate transition matrix.

The second method proposed in [MAR84] eliminates some of the disadvantages of the above method, by computing the total transition probabilities among tangible states only. The method is described as follows:

Let S = state space of the SPP, $|S|=ks$

T = set of tangible states in SPP, $|T|=kt$

V = set of vanishing states in SPP, $|V|=kv$

with $S = T \cup V$, $T \cap V = \emptyset$, and $ks = kt + kv$.

Disregarding for the time being the concept of time, and focusing attention on the set of states in which the process is led because of a transition out of a given state, it is observed that a stationary embeded markov chain (EMC) can be recognized within the SPP. The transition probability of this EMC can be written as follows:

$$U = A + B = \begin{matrix} kv \\ \\ \end{matrix}\begin{bmatrix} C & D \\ \emptyset & \emptyset \end{bmatrix} + \begin{bmatrix} \emptyset & \emptyset \\ E & F \end{bmatrix}\begin{matrix} kv \\ kt \end{matrix}$$

The elements of matrix A, which represent the probability that the process will go to a vanishing state (C) or to a tangible state (D) given that it is at a vanishing state, can be obtained using the switching disributions of random switches. And the elements of B, which represent the transition probabilities given that the process is in a tangible state, can be obtained using the firing rates of timed transitions as in relation 5.1.1. The transition probability matrix $Q = [q_{ij}]$ between tangible states only can be computed as follows:

$$q_{ij} = f_{ij} + \sum_{r \in V} e_{ir} \Pr[r \rightarrow j], \qquad i,j \in T,\ r \in V \qquad (5.1.2)$$

where f_{ij} is the transition probability from tangible state i to tangible state j, e_{ir} is the transition probability from i to a vanishing state r, and $\Pr[r \rightarrow j]$ represents the probability that the SPP moves from the state r to the state j in an arbitrary number of steps following a path through vanishing states only. The probabilities of reaching tangible states in exactly k steps of vanishing states starting from a vanishing state are given by

$$G^k = \sum_{h=\emptyset}^{K} C^h D \qquad (5.1.3)$$

The irreducibility property of the SPP insures that the spectral radium of the matrix C is less than one. This implies that the limit of the sum $\lim_{k \rightarrow \infty} \sum_{h=\emptyset}^{K} C^h$ exists, and is finite. Equation (1.2) in the matrix form becomes

$$Q = F + E\, G^{\infty} \qquad (5.1.4)$$

where, $$G^{\infty} = \begin{cases} \sum_{h=\emptyset}^{k_o} C^h D \text{ , where } C^{h=\emptyset} \quad h > k_o \\ (I - C)^{-1} \text{ , since this equals to } \sum_{h=\emptyset}^{\infty} C^h D \end{cases}$$

which corresponds to cases where there exist no loops among vanishing states, and cases where such loops exist, respectively.

The solution of the system of linear equations $Y = Y.Q$, can be interpreted in terms of the number of transitions performed by the EMC observing that

$1/y_i$ = E{ number of transitions performed by the EMC to return to state i }

Selecting state i as a reference state,

Let $V_{ij} = y_j/y_i$ = E{ number of visits to state j between two subsequent visits to state i }

The computation of steady state probability distribution of the SPP can be obtained reintroducing the concept of time by means of the average sojourn time in each state (STi , iϵT) as follows:

Let Hi = set of timed transitions enabled at state i

then $STi = 1/\sum_{k \epsilon Hi} r_k$, is the average sojourn time for state i.

The amount of time spent by the SPP to return to state i is $W_i = \sum_{j \epsilon T} V_{ij}\ STi$, where V_{ij} is considered to be the mean amount of time spent by the SPP in state j during a cycle. The average fraction of time spent by the SPP in each of its states is given by

$$P_j = V_{ij}\ STi\ /\ W_i \quad , \quad j \epsilon S$$

which is the steady state probability distribution of the SPP.

The advantage of the above method over the first method is that it reduces the impact of the size of the set of vanishing

states on the complexity of the solution from O(ks**3) in the first method to O(kt**3) + O(kv**3), where ks = kt + kv.

Appart from the fact that SPP must be irreducible, the above method, however, have a serious limitation. It implicitly assumes that the steady state probabilities of all markings that enable immediate transitions are zero. This limitation will be demonstrated by the following example.

Example 5.1 :

Consider the GSPN in figure 5.1, t_1 and t_2 are timed transitions with rates r_1 and r_2 respectively. The rest of the transitions are immediate transitions. The reachability set S with one token in the network is

1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

Solving for the steady state probability distribution of the above states using the first method, where we assume that the firing rates of all immediate transitions is x, the rate transition matrix of the corresponding M.C. is,

$$A = \begin{bmatrix} -x & x & 0 & 0 \\ x & -(x+r1) & r1 & 0 \\ 0 & 0 & -x & x \\ r2 & 0 & x & -(x+r2) \end{bmatrix}$$

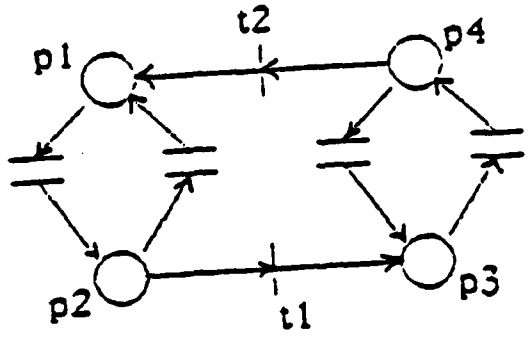


Figure 5.1

The S.S. probabilities P = [P1 P2 P3 P4] are obtained by solving

$P\,A = 0$, with $\sum_{i=1}^{4} Pi = 1$

Let $d = 2\,r_1/x + 2(1+r_1/r_2) + r_1 r_2/x$

then,

$P1 = (1+r_1/x)/d$, $P2 = 1/d$, $P3 = (1+r_2/x)\; r_1/(r_2\,d)$,

and $P4 = r_1\,/(r_2\,d)$.

In the limit as $x \to \infty$, we have

$P1 = P2 = 1/2\,.\; r_2/(r_1+r_2)$, and $P3 = P4 = 1/2.\; r_1/(r_1+r_2)$

Using the second solution method, although the GSPN and the corresponding SPP is irreducible, it is treated by this method as if it is reducible to two ergodic classes: states 1 and 2 as the first, and states 3 and 4 as the second class. Therefore, the method is not applicable in this case, and in any GSPN where ergodic instantaneous markings exist.

To demonstrate the importance of the above class of GSPNs, consider the matrix A in the above example. If every element in A is divided by x, then

$A = A(p) = A0 + p\,A1$, where $p = 1/x$,

$$A0 = \begin{vmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \end{vmatrix}, \text{ and } A1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -r1 & r1 & 0 \\ 0 & 0 & 0 & 0 \\ r2 & 0 & 0 & -r2 \end{bmatrix}$$

The steady state probability distribution of the GSPN can be obtained from those of the above M.C. be letting $p \to 0$. Clearly the above M.C. is singularly perturbed since rank A(p) > rank A(0). Therefore, a GSPN is equivalent, in the sense of steady state probability distributions to a perturbed SPN with rare

transitions modeled by a small parameter p in the limit when $p \dashrightarrow \emptyset$. And the class of GSPNs, equivalent at the limit to singularly perturbed SPNs, have an important role in the hierarchical aggregation of the later. The hierarchical aggregation of SPNs will be described in Chapter 7.

In the next and subsequent sections we will introduce a more general solution method that will alleviate the computational and numerical disadvantages of the first method by eliminating the set of vanishing states, and generalize the second method. The method that will be discussed is based on charactrizing the GSPN time behaviour by a stochastically discontinuous finite state Markov process, which is a special SPP. The properties of this process will be discussed in detail in the following section.

5.2 Stochastically Discontinuous Finite State Markov Process

The stochastically discontinuous, continuous-time, finite state markov process is a process $\{x(t),\ t \geq \emptyset\}$ that may undergo an infinite number of transitions in finite time intervals. Such processes violate the continuity condition

$$\lim_{t \dashrightarrow \emptyset} Pr\{x(t) = X(\emptyset)\} = 1$$

They were first analyzed in [DOO42, and DYN65], but were considered pathological from an application viewpoint, and since then stochastic continuity has been a standard assumption in the literature. Coderch [COD83b] has recognized that stochastically discontinuous processes are obtained as limits of markov processes with transition rates of different orders of magnitude, and that the stochastic discontinuity property has a natural and important interpretation in this context.

Stochastically discontinuous FSMP's (SDMP) are

characterized as follows:

Let $\{X(t),\ t \geq \emptyset\}$ be a FSMP taking values in a finite state space $E = \{e1,e2,\ldots,en\}$. This process is completely described by its transition probability matrix $P(t)$ whose elements are

$$p_{ij}(t) = Pr\{X(t) = j \ / \ X(\emptyset) = i\}, \quad i,j \in E,\ t \geq \emptyset.$$

and satisfies the following conditions:

i) $P(\emptyset) = I$, ii) $P(t) > \emptyset$, iii) $P(t) \cdot 1^{+} = 1^{+}$, and

iv) $P(t)\,P(s) = P(t+s) \quad t,s \geq \emptyset$, $1^{+T} = [1\ 1\ \ldots\ldots 1]$.

It is known that $P(t)$ is continuous for $t > \emptyset$, and the limit $\lim_{t \to \emptyset} P(t) = Z$ always exists. If Z is the identity matrix then the process $x(t)$ is called stochastically continuous, otherwise it is stochastically discontinuous with the following transition probability matrix:

Theorem 2.1: If $P(t)$ is the transition probability matrix of a SDMP then,

$$P(t) = Z \exp\{A\,t\} \qquad t > \emptyset \qquad (5.2.1)$$

for a pair of matrices Z, A satisfying:

i) $Z > \emptyset$, $Z.1^{+} = 1^{+}$, $Z^2 = Z$; ii) $Z.A = A.Z = A$;

iii) $A.1^{+} = 1^{+}$; iv) $A + c\,Z \geq \emptyset$ for some $c > \emptyset$.

Conversely, any matrices A, Z satisfying the (i)-(iv) uniquely determine a FSMP with transition probability matrix given by (5.2.1).

Proof: [COD83b].

The matrix $z = \lim_{t \to \emptyset} P(t)$ is referred to as the ergodic projection at zero, and the matrix $A = \lim_{h \to \emptyset} (P(h)-Z)/h$ is called the infinitismal generator of $P(t)$.

The diagonal entries of the matrix Z classify the states of

the process as follows:

Definition1: A state i is called instantaneous if $z_{ii} < 1$, and regular if $z_{ii} = 1$. An instantaneous state j is called evanescent if $z_{jj} = \emptyset$.

It was proved that:

1) the sojourn time in an instantaneous state is zero with probability one (w.p.1), and in a regular state i is exponentially distributed with rate a_{ii} (diagonal entries in A).

2) Even though the duration of stays in a given instantaneous state is zero w.p.1, there is, in general, a non-zero probability of finding the process in an instantaneous state at any given time. However, the probability of finding the process in an evanescent state at any given time is zero. The evanescent states can thus be negelected in the sense that there exists a version of the process X(t) with the same finite dimensional distributions which does not take values in the set of evanescent states.

3) Z is the matrix of ergodic probabilities of a markov chain and as such it determines a partition of the state space E in terms of ergodic classes E_i , i=1,...,s, and transient states E_T, i.e., $E = (U_{i=1}^{s} E_i) \cup E_T$ (5.2.2)
this is referred to as the ergodic partition at zero. Each ergodic class E_i consists of either one element (a regular state), or several elements (instantaneous states). The set of transient states E_T characterizes the evanescent states.

The evolution of a SDMP can be thought of as follows: While in a regular state it behaves as a stochastically continuous process. Upon entering a state belonging to an ergodic class at

zero with more than one state, say, E_k, the process starts switching instantanuously among the states in E_k. The amount of time spent in E_k is exponentially distributed, and after a random stay in E_k the process jumps to some state in $E-E_k$. Evanescent states may be visited during transitions between the ergodic classes at zero.

The probabilistic properties of a SDMP are derived from its ergodic projection at zero plus an aggregated version of the process that is stochastically continuous. This can be demonstrated as follows:

<u>proposition</u> 2.1: Let Z be the ergodic projection at zero of a SDMP, then by adequate ordering of states,

$$Z = \begin{bmatrix} Z_{11} & \emptyset & \cdots & & \emptyset \\ \emptyset & Z_{22} & \cdots & & \emptyset \\ \vdots & \vdots & & & \\ & & & Z_{ss} & \emptyset \\ Z_{1,s+1} & & & Z_{s,s+1} & \emptyset \end{bmatrix} \qquad (5.2.3)$$

with $Z_{kk} = 1^+.w_k^T$, k=1,...s, for some vector $w_k \geq \emptyset$ such that $w_k^T.1^+ = 1$; and $Z_{k,s+1} = d_k.w_k^T$, k=1,...s for a set of vectors $d_k \geq \emptyset$, such that $\sum_{k=1}^{s} d_k = 1^+$.

Furthermore, define the (nxs) matrix V and the (sxn) matrix U as follows:

$$V = \begin{bmatrix} 1^+ & \emptyset & \cdots\cdots & \emptyset \\ \emptyset & 1^+ & \cdots\cdots & \emptyset \\ . & \emptyset & . & \emptyset \\ . & . & 1^+ & \emptyset \\ . & & . \; \emptyset & 1^+ \\ d_1 & d_2 & & d_s \end{bmatrix}$$

$$U = \begin{bmatrix} w_1^T & \emptyset & \cdots\cdots\cdots & \emptyset \\ \emptyset & w_2^T & \cdots\cdots & \emptyset \\ . & . & . & \emptyset \\ \emptyset & \emptyset & w_s^T & \emptyset \end{bmatrix} \qquad (5.2.4)$$

Then,

$$V \cdot U = Z \;, \quad U \cdot V = I \qquad (5.2.5)$$

Proof: Follows from the fact that Z is the matrix of ergodic probabilities of a markov chain. The vector w_k is the vector of steady state probabilities of a chain with state space E_k and steady state transition matrix Z_{kk}. The vectors d_k are the trapping probabilities from transient states to the ergodic classes [DOOS3].

The structure of (5.2.3) makes explicit the ergodic partition at zero. (2.4) is called the canonical product decomposition of Z. Also U and V satisfy the following

$U \cdot 1^+ = 1^+$, $V \cdot 1^+ = 1^+$, $U \cdot Z = U$, and $Z \cdot V = V$

Theorem 2.2: Let $P(t) = Z \exp\{A\, t\}$ be the transition probability matrix of a SDMP X(t) taking values in E={e1,e2,...,en} and let s be the number of ergodic classes at zero. Let $Z = V \cdot U$ be the canonical product decomposition of Z, then

$$P'(t) = U\, P(t)\, V = \exp\{U\, A\, V\, t\} \;, \text{ for all } t > \emptyset \qquad (5.2.6)$$

is the transition probability matrix of a stochastically continuous FSMP taking values in E'={e1',e2',...,es'}, and

$$P(t) = V\ P'(t)\ U \quad \text{for all } t > \emptyset \qquad (5.2.7)$$

Proof: [COD83b]

Equation (2.6) can be interpreted as performing an aggregation operation that masks the stochastically discontinuous nature of P(t). Also equation (2.7) can be interpreted as follows:

$$Pr\{X(t)=ei \ / \ X(\emptyset)=ej\} = w_{1i} \cdot Pr\{X'(t)=el' \ / \ X'(\emptyset)=ep'\}$$
$$, ej \in E_p \ , \ ei \in E_l$$

where w_{1i} is the component of the steady state probability vector w_1 corresponding to ei. That is, the transitions between the ergodic classes E_i are governed by the aggregated process, while once in one of the classes E_l, the probabilities w_1 are immediatly established due to the instantaneous nature of transitions.

It should be noted here that the above aggregation is exact, i.e., there is no approximation involved whatsoever, whereas the aggregation described in the previous chapter was approximate due to the fact that the transitions were not quite instantaneous.

Corollary 2.1: The rate transition matrix A' of the aggregated process X'(t), which is the infinitismal generator of P'(t), is given by,

$$A' = U\ A_1\ V \qquad (5.2.8)$$

where A_1 is the matrix of transition rates of the process X(t) when all instantaneous transitions have been removed.

Proof: Follows from theorem 2.2 above, and the theory of

singularly perturbed FSMP's presented in the previous chapter.

Example 5.2: Consider the GSPN in figure 1.1 of the previous section. The graph of the GSPN represents the transition diagram of a SDMP with state space E={e1,e2,e3,e4} represented as follows

e1	1	0	0	0
e2	0	1	0	0
e3	0	0	1	0
e4	0	0	0	1

Clearly the ergodic partition at zero is

E_1={e1,e2} , and E_2={e3,e4}

and $Z = \begin{bmatrix} z_{11} & 0 \\ 0 & z_{22} \end{bmatrix}$, where $z_{11}=z_{22}= \begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix}$,

since $w_1^T = w_2^T = [1/2 \quad 1/2]$

then $V = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$, and $U = \begin{bmatrix} 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 \end{bmatrix}$

The rate transition matrix A' is given by

$A' = U A_1 V$, where $A_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -r_1 & r_1 & 0 \\ 0 & 0 & 0 & 0 \\ r_2 & 0 & 0 & -r_2 \end{bmatrix}$

then $A' = \begin{bmatrix} -1/2\ r_1 & 1/2\ r_1 \\ 1/2\ r_2 & -1/2\ r_2 \end{bmatrix}$

Solving the aggregated process for the steady state probabilities of the ergodic classes, we have

$P(E_1) = r_2 / (r_1+r_2)$, $P(E_2) = r_1 / (r_1+r_2)$

and the steady state probabilities of the SDMP are evaluated by

$\begin{bmatrix} P(e1) \\ P(e2) \end{bmatrix} = w_1 \cdot P(E_1)$, and $\begin{bmatrix} P(e3) \\ P(e4) \end{bmatrix} = w_2 \cdot P(E_2)$

therefore,

$P(e1)=P(e2)= 1/2\ r_2/(r_1+r_2)$, and $P(e3)=P(e4)= 1/2\ r_1/(r_1+r_2)$ which is the same result obtained before using the first method in section 1.

5.3 Evaluation of the GSPN steady state probability distribution

In this section a solution method for the steady state probability distribution of the GSPN will be presented. This method is based on characterizing the time behaviour of a GSPN as a SDMP.

Theorem 3.1: The marking sequence of a live and k-bounded GSPN forms a SDMP.

proof: Let S = {M1,M2,...,Mn} be a reachability set of a live and bounded GSPN with an initial marking M1. Since the GSPN is live, then there exist no marking in S at which all transitions are disabled.

Let {X(t), t>=0} be a stochastic process with a finite state space E = {1,2,...,n}, such that

1- There exist a one to one mapping F between the elements of E and the elements of S, i.e., for each Mi ϵ S, there exist a corresponding state i ϵ E, such that F(Mi) = i ,

2- For each Mi,Mj ϵ S, where Mj is reachable from Mi by the firing of a transition enabled by Mi, there exists a transition in X(t) from the corresponfing state i to j, and

3- For all i ϵ E, the soujorn time of i is equal to that of Mi, i.e,

$$Pr[\ X(w) = i,\quad w \in [0,t]/\ X(0) = i] = \exp\{-y_i\ t\}$$

where $yi = \sum_{j=1}^{K} rj$, and rj is the rate of transition tj enabled at Mi, k>=1. If any one of these transitions is an immediate transition, the above probability will be zero, since the rate of such a transition is infinite. Therefore, the soujorn time of state i is either zero if Mi enables any immediate transition , or exponentially distributed if Mi enables only timed transitions.

The state space E of the above process can be partitioned into a set of instantaneous states, and a set of regular states with exponentially distributed soujorn time. Clearly, if all states are regular, then X(t) is a finite state stochastically continuous Markov process. From the theory described in the previous section, the existence of instantaneous states results in a Markov process, The transition probability matrix of which P(t) is discontinuous at t=0, i.e., from theorem 2.1

$P(t) = Z \exp\{A\ t\}$, $t > 0$, $P(0) = I$, and $Z = \lim_{t \to 0} P(t)$

If all states are regular, then for each state i , zii = 1, and Z = I. If there exist an instantaneous state i, then zii < 1, and the process is stochastically discontinuous. #

The above theorm establishes the fact that the steady state probability distributtion of the GSPN markings can be obtained by solving the corresponding SDMP. As in the previous section, we need to obtain the matrices U, V, and A', which fully characterize the time behaviour of the process. These matrices are obtained as follows:

From the reachability graph analysis [NAT 80, FLO 84,85, CHI 85] of the GSPN under immediate transitions only, the reachability set S can be partitioned into two subsets S_1 and S_2.

The subset S_2 contains markings which enable any immediate transition, and markings reachable by the firing of an immediate transition. The subset S_1 contains all other markings. Furthermore, S_2 is partitioned into several subsets S2i, i=1,..,g, and a subset S2T. This partition corresponds to the state space partition into ergodic classes at zero expressed in equation (2.2) as follows:

$$E = S = S_1 \cup S_2 \qquad (5.3.1)$$

where, $S_1 = \cup_{i=1}^{K} ei'$, $S_2 = (\cup_{i=1}^{g} S2i) \cup S2T$, and $k + g = s$ which is the total number of ergodic classes at zero. Each S2i consists of one ergodic class which may contain one marking that absorb the process under immediate transitions, or several markings reachable from one another by immediate transitions. Each ei' consists of one marking. And the set S2T consists of transient markings.

The reason for the above modification of the partition of ergodic classes at zero is to allow a more straightforward construction of the matrices U and V, which can then be partitioned as follows:

$$V = \begin{matrix} S1 \\ S2 \end{matrix} \begin{bmatrix} I_k & \emptyset \\ \emptyset & K' \end{bmatrix} \quad , \text{ and } \quad U = \begin{matrix} \\ \end{matrix} \begin{matrix} S1 & S2 \\ \end{matrix} \begin{bmatrix} I_k & \emptyset \\ \emptyset & K'' \end{bmatrix} \qquad (5.3.2)$$

where I_k is an identity matrix of dimension k, K' is an (n-k)xg matrix, and K" is a gx(n-k) matrix given by

$$K' = \begin{array}{c} \\ S21 \\ S22 \\ \cdot \\ \cdot \\ S2g \\ S2T \end{array} \begin{bmatrix} 1^+ & & & \\ & 1^+ & & \\ & & & \\ & & & 1^+ \\ d_1 & d_2 & \cdot & d_g \end{bmatrix}$$

$$K'' = \begin{array}{c} \\ S21 \\ S22 \\ \\ \\ S2g \\ S2T \end{array} \begin{bmatrix} w_1^T & & & \emptyset \\ & w_2^T & & \emptyset \\ & & & \\ & & w_g^T & \emptyset \end{bmatrix}$$

where d_i is a vector of trapping probabilities from transient markings to the ergodic class in S2i. And w_i as before is the steady state probability vector of a markov chain with state space S2i.

The vectors of trapping probabilities can easily be obtained as follows: consider an absorbing MC with a state space defined by the union of the set S2T, and a set of g absorbing states each of which corresponds to an ergodic class S2i, i=1,..,g. The transition probability matrix PT of this MC is given by

$$PT = \begin{array}{c} \\ S2T \end{array} \begin{bmatrix} I_g & \emptyset \\ Y & X \end{bmatrix}$$

Where I_g is an identity matrix of dimension g, y_{ij} is the transition probability to any state in ergodic class j from a state i in S2T, and x_{ij} is the transition probability to state j in S2T from state i in S2T. The vectors of trapping probabilities are then computed as follows [KEM 60],

$$[d1\ d2\ .\ .\ dg] = (I-X)^{-1} \cdot Y \qquad (5.3.3)$$

To obtain the aggregated matrix A', consider now the GSPN under timed transitions only. The matrix A_1, which is the matrix of transition rates between markings that enables timed transitions, is also partitioned as follows:

$A_1 = \begin{bmatrix} A'' & B'' \\ C'' & D'' \end{bmatrix}$, where A" is a kxk matrix with off-diagonal elements representing transition rates between marking that belong to the set S_1, B" is an kx(n-k) matrix with elements representing the transition rates from markings that belong to S_1 to markings that belong to S_2, and such that the diagonal elements of A" is the negative of the sum of the off-diagonal elements of each row in A" plus the elements in the corresponding row in B". Also D" is an (n-k)x(n-k) matrix with off-diagonal elements representing transition rates betweem markings that belong to S_2, and C" is an (n-k)xk matrix of transition rates from markings in S_2 back to markings in S_1, such that the diagonal elements of D" is the sum of the off-diagonal elements of D" and the elements of C" for each raw.

Using equation (2.8), the sxs matrix A' is obtained as follows:

$$A' = U A_1 V = \begin{bmatrix} A'' & B''K' \\ K''C'' & K''D''K' \end{bmatrix} \qquad (3.4)$$

Which can then be solved for the steady state probabilities of the ergodic classes at zero P(ei'), i=1,...k, and P(S2i), i=1,...,g. The steady state probability distribution of the GSPN markings can then be evaluated from,

P(Msi) = P(ei) , i=1,.....,k , where Msi are markings that belong to S_1, and

$$\begin{bmatrix} P(M1^i) \\ \cdot \\ \cdot \\ P(Mj^i) \end{bmatrix} = w_i^T P(S2i)$$ for the j markings that belong to S2i, i=1,.....,g.

3.4 Examples

1) Consider the simple GSPN shown in figure 4.7, with one token initially in p_1, the reachability set is

M1	1	0	0	0	0
M2	0	1	1	0	0
M3	0	0	1	1	0
M4	0	1	0	0	1
M5	0	0	0	1	1

Under immediate transition t_3, we have M5 --> M1, therefore

S_1 = {M2,M3,M4} = {e1,e2,e3} , S_{21} = {M1}, and

S2T = {M5}.

Then $V = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, $U = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$

i.e., $K' = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, $K'' = w_1^T = [\ 1 \quad 0\ ]$

also $A'' = \begin{bmatrix} -(r1+r2) & r1 & r2 \\ 0 & -r2 & 0 \\ 0 & 0 & -r1 \end{bmatrix}$, $B'' = \begin{bmatrix} 0 & 0 \\ 0 & r2 \\ 0 & r1 \end{bmatrix}$

and $D'' = \begin{bmatrix} -r0 & 0 \\ 0 & 0 \end{bmatrix}$, $C'' = \begin{bmatrix} r0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

and using (3.4) the aggregated transition matrix is given by

$$A' = \begin{bmatrix} -(r1+r2) & r1 & r2 & 0 \\ 0 & -r2 & 0 & r2 \\ 0 & 0 & -r1 & r1 \\ r0 & 0 & 0 & -r0 \end{bmatrix}$$

2) Consider the GSPN in figure 5.3. t_5 and t_6 are immediate transitions that form a random switch, and fire with probability P(5) and P(6), respectively. The reachability set S is

M1	1	0	0	0	0	0	0
M2	0	1	0	0	0	0	0
M3	0	0	1	1	0	0	0
M4	0	0	0	1	1	0	0
M5	0	0	1	0	0	1	0
M6	0	0	0	0	1	1	0
M7	0	0	0	0	0	0	1

Considering immediate transitions only we have

We can clearly distinguish two ergodic states M1 and M3. Therefore,

S21 = {M3} , S22 = {M1}, and S2T = {M2,M6,M7}

where M6, M7, and M2 are evanescent states. To obtain the trapping probability vectors from these states to S21 and S22, we have,

$$PT = \begin{array}{c} \\ M3 \\ M1 \\ M2 \\ M6 \\ M7 \end{array} \begin{array}{c} \begin{array}{cc} \quad & \quad \end{array} \\ \left[\begin{array}{cc|ccc} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & P(6) & P(5) & 0 & 0 \end{array}\right] \end{array}$$

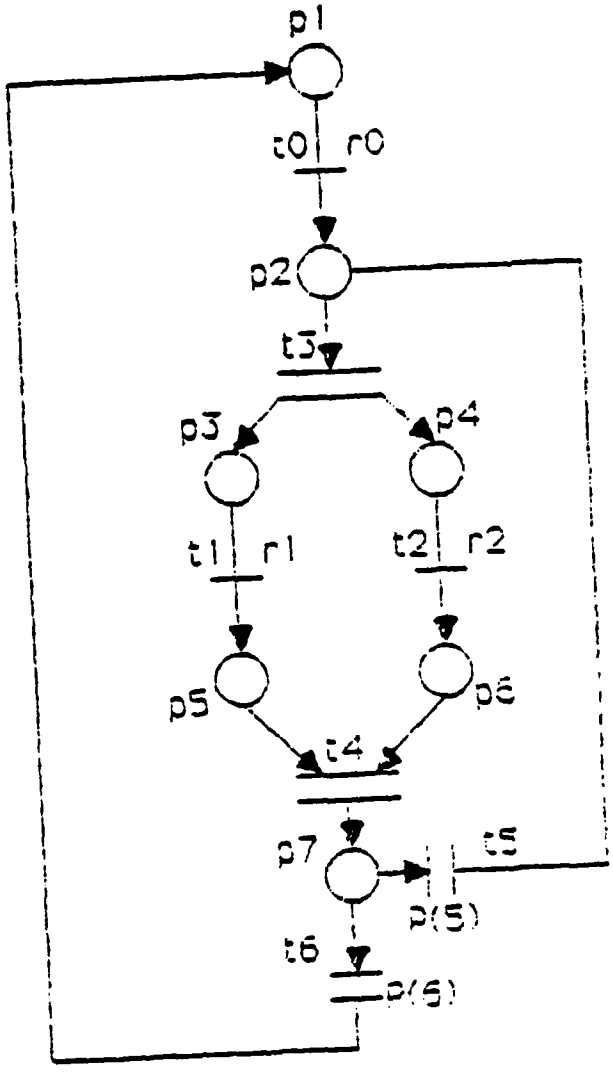


Figure 5.3

,and from equation (3.3) $[d1\ d2\ ] = \begin{bmatrix} 1 & \emptyset \\ P(5) & P(6) \\ P(5) & P(6) \end{bmatrix}$

$$K' = \begin{matrix} M3 \\ M1 \\ M2 \\ M6 \\ M7 \end{matrix}\begin{bmatrix} 1 & \emptyset \\ \emptyset & 1 \\ 1 & \emptyset \\ P(5) & P(6) \\ P(5) & P(6) \end{bmatrix}, \text{ and } K'' = \begin{bmatrix} 1 & \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & 1 & \emptyset & \emptyset & \emptyset \end{bmatrix}$$

Now considering timed transitions, we have

$$A'' = \begin{matrix} M4 \\ M5 \end{matrix}\begin{bmatrix} -r2 & \emptyset \\ \emptyset & -r1 \end{bmatrix}, \quad B'' = \begin{bmatrix} \emptyset & \emptyset & \emptyset & r2 & \emptyset \\ \emptyset & \emptyset & \emptyset & r1 & \emptyset \end{bmatrix},$$

$$C'' = \begin{matrix} M3 \\ M1 \\ M2 \\ M6 \\ M7 \end{matrix}\begin{bmatrix} r1 & r2 \\ \emptyset & \emptyset \\ \emptyset & \emptyset \\ \emptyset & \emptyset \\ \emptyset & \emptyset \end{bmatrix}, \text{ and } D'' = \begin{bmatrix} -(r1+r2) & \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & -r\emptyset & r\emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset & \emptyset \end{bmatrix}$$

then $B''K' = \begin{bmatrix} P(5)r2 & P(6)r2 \\ P(5)r1 & P(6)r1 \end{bmatrix}$, $K''C'' = \begin{bmatrix} r1 & r2 \\ \emptyset & \emptyset \end{bmatrix}$, and

$K''D''K' = \begin{bmatrix} -(r1+r2) & \emptyset \\ r\emptyset & -r\emptyset \end{bmatrix}$, therefore, we have

$$A' = \begin{bmatrix} -r2 & \emptyset & P(5)r2 & P(6)r2 \\ \emptyset & -r1 & P(5)r1 & P(6)r1 \\ r1 & r2 & -(r1+r2) & \emptyset \\ \emptyset & \emptyset & r\emptyset & -r\emptyset \end{bmatrix}$$

3) Consider the GSPN in figure 3.4, transitions t_2 and t_3 can be simultaneously enabled. Therefore, the probabilities P(2) and

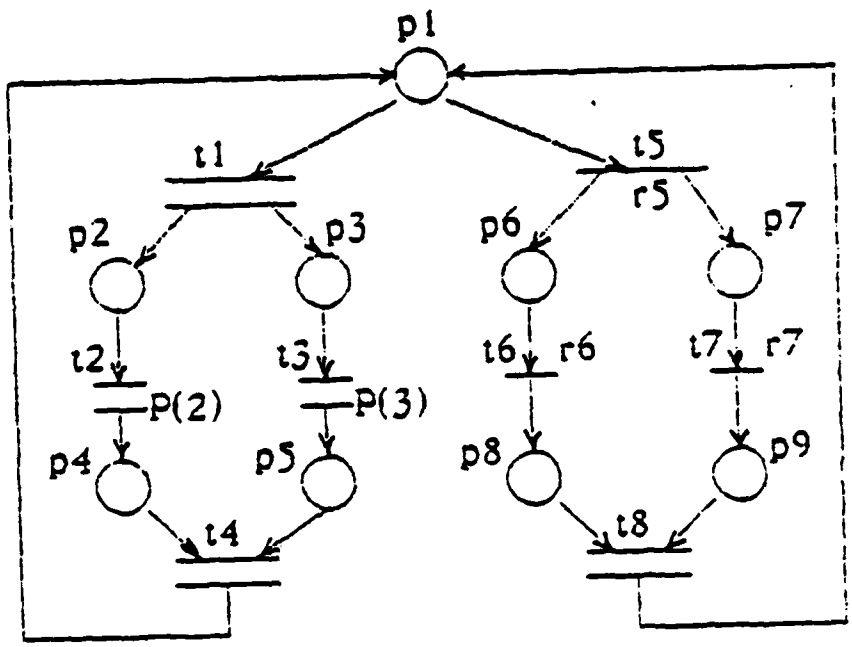
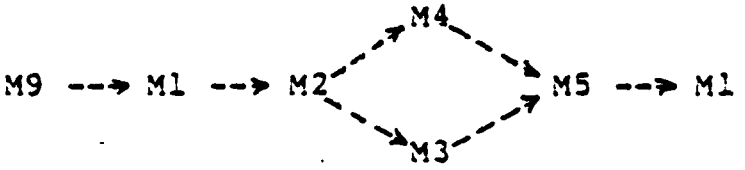


Figure 5.4

P(3) are assigned to determine wich one will fire first. The reachability set is given by

M1	1	0	0	0	0	0	0	0	0
M2	0	1	1	0	0	0	0	0	0
M3	0	0	1	1	0	0	0	0	0
M4	0	1	0	0	1	0	0	0	0
M5	0	0	0	1	1	0	0	0	0
M6	0	0	0	0	0	1	1	0	0
M7	0	0	0	0	0	0	1	1	0
M8	0	0	0	0	0	1	0	0	1
M9	0	0	0	0	0	0	0	1	1

Again considering immediate transitions only, we have



Therefore, there is one ergodic class given by

S21 = {M1,M2,M3,M4,M5}, where M9 is a transient state. Solving the above markov chain, which has a transition probability matrix P given by

$$P = \begin{array}{c} \\ M1 \\ M2 \\ M3 \\ M4 \\ M5 \end{array} \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & P(2) & P(3) & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

for the steady state probabilities we get,

$w_1^T = [1/4 \quad 1/4 \quad P(2)/4 \quad P(3)/4 \quad 1/4\ ]$.

Then, $K'^T = [1\ 1\ 1\ 1\ 1\ 1]$, and $K'' = w_1^T$

Considering timed transitions, we get,

$$A'' = \begin{matrix} \\ M6 \\ M7 \\ M8 \end{matrix}\begin{bmatrix} -(r6+r7) & r6 & r7 \\ 0 & -r7 & 0 \\ 0 & 0 & -r6 \end{bmatrix}, \quad B'' = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & r7 \\ 0 & 0 & 0 & 0 & 0 & r6 \end{bmatrix}$$

$$C'' = \begin{matrix} M1 \\ M2 \\ M3 \\ M4 \\ M5 \\ M9 \end{matrix}\begin{bmatrix} r5 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad D'' = \begin{bmatrix} -r5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Then,

$$B''K' = \begin{bmatrix} 0 \\ r7 \\ r6 \end{bmatrix}, \quad K''C'' = [1/4.r5 \quad 0 \quad 0] \text{ , and } K''D''K' = [-1/4.r5]$$

Therefore,

$$A' = \begin{bmatrix} -(r6+r7) & r6 & r7 & 0 \\ 0 & -r7 & 0 & r7 \\ 0 & 0 & -r6 & r6 \\ 1/4.r5 & 0 & 0 & -1/4.r5 \end{bmatrix}$$

CHAPTER 6

TECHNIQUES FOR REDUCING ANALYSIS COMPLEXITY

6.1 Introduction

The method described in the previous chapter is valid for the analysis of general GSPNs. However, the analysis can be quite complicated for GSPNs with large state spaces. For example, consider the GSPN in example 1 of section 5.4 in the previous Chapter, with one token initially in p1, there were 5 feasible states for the network. If, however, we added k tokens in p1, the number of states will be in the order of 5^k. Therefore even with such a simple GSPN, an explosion of the state space can make the analysis very complicated.

In this chapter, rather than describing the stochastic behaviour of a GSPN by a SDMP, which is then analyzed from its projection at zero plus an aggregated version of it represented by the rate transition matrix A', we attempt to do such aggregation or reduction directly at the GSPN level.

The analysis in this chapter will be restricted to a class of GSPNs that inherits the structure of restricted PNs. Such PNs will be defined in the following section, and some of its important properties will be developed.

6.2 Restricted Petri Nets

Definition 1: A Petri Net PN = (P,T,I,O), with an initial marking M1 and a reachability set S, is called a restricted PN if all arcs have a weight of one, i.e., the input and output functions

are such that, I: PXT --> {0,1} , and O: TXP --> {0,1}, and for any transition in T the set of input places and the set of output places are disjoint (self-loop-free).

We will also assume that the PN is live and bounded, i.e., for every ti $\in$ T and for all Mk $\in$ S there exists a transition firing sequence starting at Mk and ending at a marking that enables ti. An important property of restricted PNs is established by the following theorem,

Theorem 1:(superposition theorem)

For any restricted PN, let S1,S2,..,Sk be reachability sets obtained from the different initial markings M11,M12,..,M1k, respectively. Then for an initial marking M1' = M11+M12+...+M1k, which gives a reachability set S',

if $Mr' = Mj1^1+Mj2^2+\ldots+Mjk^k$ (6.2.1)

,where $Mji^i \in Si$, i=1,..,k

then Mr' $\in$ S', i.e., S' $\supset$ (s1+s2+...+sk).

Moreover, the above condition becomes necessary and sufficient if for every initial marking M1i, i=1,...k, the PN is live.

To prove the above theorm, we need to introduce the following definitions.

Definition 2: for a restricted PN = (P,T,I,O),

where P = {p1,p2,..,pn}, T = {t1,t2,..,tm},

I:PXT-> {0,1} , and O:TXP -> {0,1}, with an initial marking M1 and reachability set S, then for any Mk,Mk-1$\in$ S, such that Mk is immediatly reachable from Mk-1 by the firing of transition tj,

$$Mk = Mk-1 + D^T Uj \qquad (6.2.2)$$

where Mk and Mk-1 are nx1 column vectors, Uj is an mx1 column vector with exactly one nonzero entry in the position corresponding to transition tj, and D is an mxn matrix called transition to place incidence matrix defined as follows:

$D = - D^{-T} + D^{+}$, where $D^{-} = \{dij^{-}\} = \{I(pi,tj)\}$,

and $D^{+} = \{dij^{+}\} = \{O(ti,pj)\}$.

Therefore, the entries of matrix D; dij, are 1, -1, or Ø if transition ti has an outgoing arc to place pj, an incoming arc from place pj, or no arc between them, respectively.

Equation 6.2.2 is the matrix form of equation 2.2.1 in Chapter 2 [MUR77, PET81]. For example, for M1 and M2 in exmaple 1 of section 5.4,

$$\begin{bmatrix} Ø \\ 1 \\ 1 \\ Ø \\ Ø \end{bmatrix} = \begin{bmatrix} 1 \\ Ø \\ Ø \\ Ø \\ Ø \end{bmatrix} + \begin{array}{c} \\ p1 \\ p2 \\ p3 \\ p4 \\ p5 \end{array} \begin{array}{c} \begin{array}{cccc} t1 & t2 & t3 & t4 \end{array} \\ \begin{bmatrix} -1 & Ø & Ø & 1 \\ 1 & -1 & Ø & Ø \\ 1 & Ø & -1 & Ø \\ Ø & 1 & Ø & -1 \\ Ø & Ø & 1 & -1 \end{bmatrix} \end{array} \begin{bmatrix} 1 \\ Ø \\ Ø \\ Ø \\ Ø \end{bmatrix}$$

The above can be extended to a sequence of transition firings as follows:

Definition 3: for any PN with an initial marking M1, incidence matrix D, and reachability set S, if Mk $\notin$ S, then

$$Mk = M1 + D^{T} U_{1,k} \qquad (6.2.3)$$

where $U_{1,k}$ is an mx1 column vector called the firing vector, the ith element of which is the number of times transition ti fires in the transition firing sequence (tj1,tj2,..,tjk) starting from M1 and ending at Mk, i.e.,

$$U_{1,k} = \sum_{i=1}^{k} Uji$$

roof of theorem 1:

Let $Mr' = Mj1^1 + .. + Mjk^k$,

for any $Mji^i \in Si$, i=1,..,k.

By 6.2.3, $Mji^i = Mli + D^T U_{1,ji}$, i=1,..,k.

then $Mr' = \sum_{i=1}^{K} Mli + D^T U$, where $U = \sum_{i=1}^{K} U_{1,ji}$

$= Ml' + D^T U$

Since Ml' is the initial marking for the reachability set S', and all the transition sequences in U are defined in Ml', then again by 6.2.3, Mr' is reachable from Ml', i.e. $Mr' \in S'$.

For the second part of the theorem, the above proves sufficiency, the proof of necessity is done by contradiction as follows:

Let $Mr' \notin S'$, and $Mr' =/= Mj1^1 + ... + Mjk^k$, for any $Mji^i \in Si$, then, $Mr' = Ml' + D^T U = Ml1 + Ml2 + .. + Mlk + D^T U$,

and since $Mr' =/= \sum_{i=1}^{K} Mji^i$, then

there exists no U1,ji, such that $U = \sum_{i=1}^{K} U1,ji$ that is defined by a sequence of transition firing from Mli, then

$Mr' = \sum_{i=1}^{K} Mmi^i + D^T U'$, where $Mmi^i \in Si$,

and U' is a firing vector of transitions not enabled by any Mmi^i, i=1,...,k. Since the firing vector U is defined for Ml', then there exist at least two markings, from the above set of markings, that can be added together to enable a transition in U'. Therefore, there exist at least one initial marking from the set {Ml1,Ml2,...,Mlk} for which the PN is not live which is a contradiction. ‡

An important consequence of the above theorem is that many important characteristics of a live restricted PN, with an initial marking Ml' and reachability S', can be studied by dividing Ml'

into several initial marking Mli with a reduced reachability sets Si, i=1,...k. And if, under each one of these initial markings, the PN is live, then the reachability set S' can be constructed by adding all possible combinations of markings in the reduced sets Si.

corollary 1: for a live restricted PN with an initial marking M1, and a reachability set S, if an initial marking M1' = k M1, for some integer k, is considered, a reachability set S' is obtained such that, for any Mi' ϵ S',

Mi' = Mj1 + Mj2 + ... + Mjk , where Mjl, l=1,..,k, are in S.

The above corollary can be used to analyze the behaviour of restricted PNs where there exists a place pl P, called the exciting place, such that, the initial marking of the PN is defined by one or more tokens in pl and zero tokens in all other places. Such PNs are particularly suitable for modeling jobs behaviour as described in the previous chapter, where the number of tokens initially in pl resembles the number of jobs that are being processed in the system (the multiprogramming level).

6.3 Reduction And Aggregation of GSPNs.

In this section, reduction and aggregation of GSPNs will be considered. By reduction we mean the elimination of immediate transitions that caused the existence of transient instantaneous markings (the steady state probabilities of which will always be zero). And by aggregation we mean, the aggregation of subnetworks consisting of immediate transitions that caused the existence of ergodic classes of instantaneous markings at time t = 0.

Figure 6.1 shows four examples of the reduction process. These examples involve subnetworks containing single input-single output, single input-multiple output, multiple input-single output, and multiple input-multiple output immediate transitions, respectively. This process is done locally in each subnetwork without affecting the rest of the network, which is a very important property. The multiple input immediate transitions, as the ones shown in figure 6.1 (c) and (d), cannot be eliminated if they are in conflict (i.e. share a common input place) with any other immediate transition. As was shown in Chapter 4, by using such conflicting multliple input immediate transitions, we are able to model queuing systems with multiple classes of customers and fixed priority queuing disciplines, which can not be modeled by SPNs. This is ofcourse due to the fact that SPNs form a subclass of GSPNs. The above reduction process will be investigated further towards the end of this section.

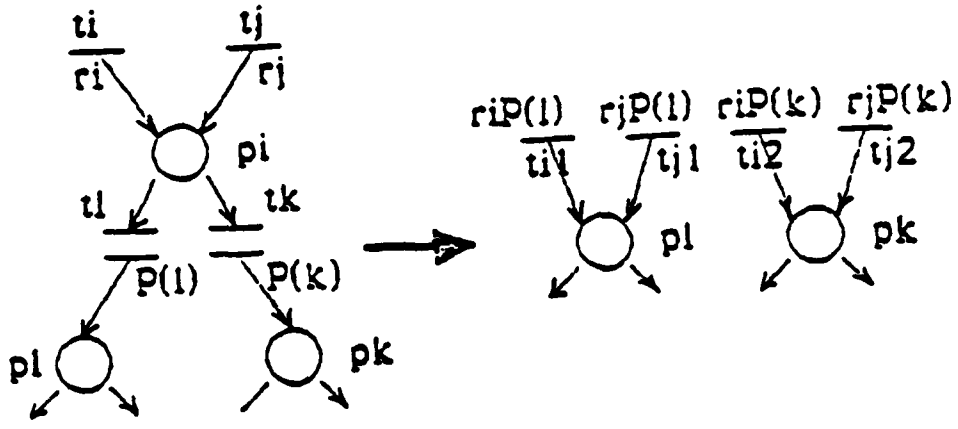
The aggregation process can be carried out on a class of subnetworks defined by the following definitions.

Definition 4: for a GSPN = (P,T,I,O), with an initial marking M1 and a reachability set S. A subnetwork N = (P1,T1,I1,O1) is defined such that, $T1 \subset T$ is a set of immediate transitions. $P1 \subset P$ is the set of input and output places of the transitions in T1, i.e., for any $pi \in P$, if and only if

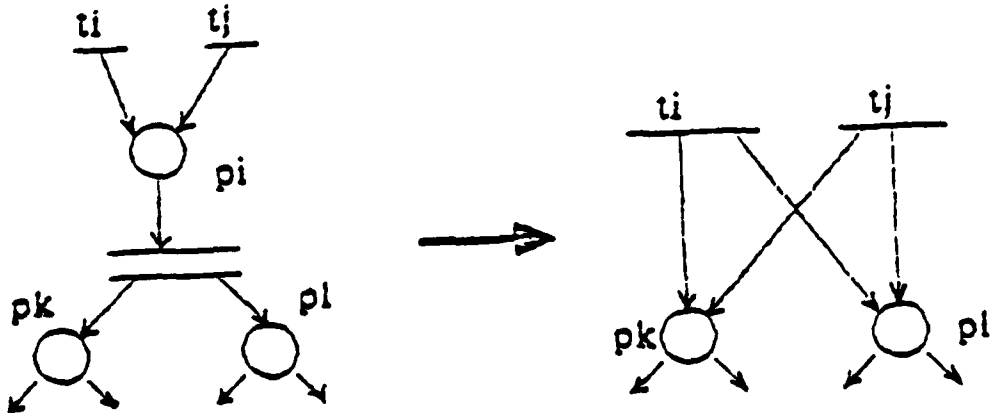
$I(pi,tj) = 1$, or $O(tj,pi) = 1$, for any $tj \in T1$,

then $pi \in P1$.

Also I1 and O1 are the input output functions I and O restricted to P1 and T1, i.e.,

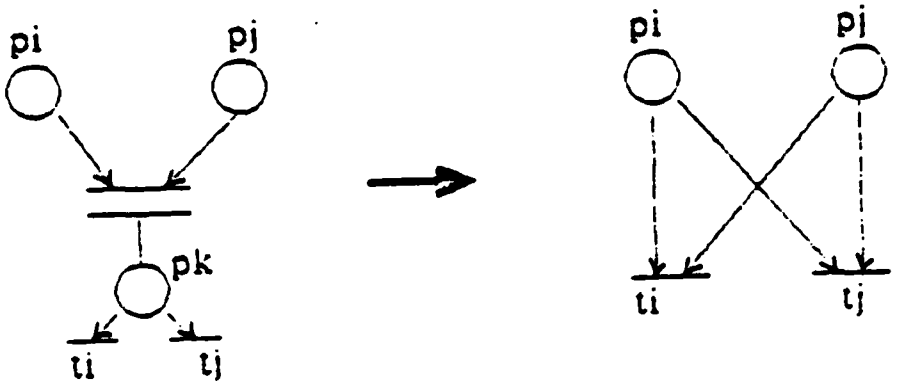

ti
tj
ri
rj
pi
tl
tk
P(l)
P(k)
pl
pk
riP(l)
rjP(l)
riP(k)
rjP(k)
til
tjl
ti2
tj2
pl
pk

(a)

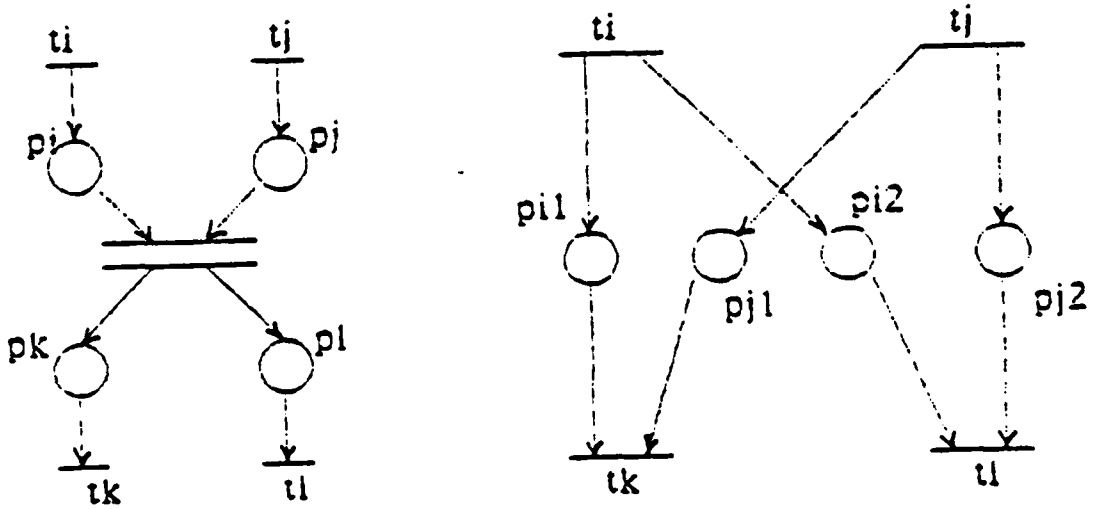

ti
tj
pi
pk
pl
ti
tj
pk
pl

(b)

Figure 6.1


pi
pj
pk
ti
tj
pi
pj
ti
tj

(c)


ti
tj
pi
pj
pk
pl
tk
tl
ti
tj
pi1
pi2
pj1
pj2
tk
tl

(d)

Figure 6.1

$$I1:P1XT1 \rightarrow \{0,1\} \text{ such that } I1(pi,tj) = \begin{cases} I(pi,tj), \text{if } (pi,,tj) \in P1XT1 \\ 0 \text{ if not} \end{cases}$$

$$O1:T1XP1 \rightarrow \{0,1\} \text{ such that } O1(tj,pi) = \begin{cases} O(tj,pi), \text{if } (tj,pi) \in T1XP1 \\ 0 \text{ if not} \end{cases}$$

Definition 5: for subnetwork N defined above, the set of places Pin $\subset$ P1 and the set of transitions Tin $\subset$ {T-T1}, are defined such that for any pi $\in$ Pin, O(tj,pi) = 1 for some tj $\in$ Tin. Also the set of places Pout $\subset$ P1 and the set of transitions Tout $\subset$ {T-T1} are defined such that, for any pi $\in$ P1, with I(pi,tj) = 1 for some tj $\in$ {T-T1}, then pi $\in$ Pout and tj $\in$ Tout.

Transitions in Tin deposit tokens into places in the set Pin of subnetwork N. And transitions in Tout remove tokens from places in the set Pout of N.

The subnetwork N defined above partitions the reachability set S into two subsets defined as follows,

Definition 6: the subnetwork N partitions the set S into two subsets S1 and S2, such that for all Mi $\in$ S1 and all pi $\in$ P1, Mi(pi) = 0, and for all Mj $\in$ S2, Mj(pi) > 0. Moreover the set S2 can also be partitioned into several subsets as follows,

$S2 = \bigcup_{i=1}^{l} S2i$, such that for any Mk,Mj $\in$ S2i, Mk(pn)=Mj(pn) for all pn $\in$ {P-P1}. Therefore, if Mj is reachable from Mk, then all transitions in the transition sequence starting at Mk and ending at Mj belong to T1.

Definition 7: The subnetwork N defined above is said to be recurrent if for each S2i, i=1,..l, and for any Mk,Mi $\in$ S2i, Mk is reachable from Mj by a finite sequence of transitions in T1.

Definition 8: The sets S2i, i=1,..,l belong to equivalent classes denoted by the sets Se1,Se2,..,Sem, where each Sei = {S2i1,S2i2,...,S2ir}, such that the sets S2ij, j=1,..,r contain exactly the same number of markings, and for any S2ij,S2in $\in$ Sei, there exist a marking Mi $\in$ S2ij and a marking Mj $\in$ S2in such that for all pk $\in$ Pl, Mi(pk) = Mj(pk).

Each of the equivalent classes defined above is obtained from a different initial marking in the subnetwork N. Therefore, if we define a marking function M^N, which the marking M restricted to the set of places Pl of subnetwork N, then the sets in each one of the classes Sei, i=1,..,m, become indistinguishable, and therefore, each of the above classes reduces to a set of markings defined on Pl and obtained from an initial marking M^N1i, i=1,..,m. These initial markings are introduced into the subnetwork by the firing of one or more transitions in Tin (the set of input transitions of N) which modify the markings of places in Pin(the set of input places of N).

For a recurrent subnetwork, and for each initial marking M^N1i, the marking sequence in the subnetwork is isomorphic to an ergodic discrete parameter Markov chain, and the steady state probability distribution of the number of tokens in each place can be obtained . However, in order to analyze a subnetwork in isolation of the rest of the network, the following locality condition must be satisfied.

Definition 9 (locality): For a subnetwork N, if the probability of firing a transition in N is dependent only on the local

markings of N, then N is said to satisfy the locality condition

Definition 10 (conservation): for a subnetwork N, and for any initial marking M^Ntli of N, if the total number of tokens in M^Nli is equal to the total number of tokens in any marking that enables an output transition in Tout, N is said to satisfy the conservation condition.

The above definition is merely stating that a conservative subnetwork is a one which does not create (or eleminate) tokens to (from) the rest of the network.

The aggregation of a subnetwork with immediate transitions is given in the following theorm,

Theorem 2: For live and bounded restricted GSPN B=(P,T,I,O) with an initial marking M1 and a set of transition firing rates R (defined for timed transitions), if there exists a sunbetwork N = (P1,T1,I1,O1) as defined in definition 4, such that,

i) The set of input places Pin contains only one element, and the set of output transitions Tout consists of timed transitions,

ii) The subnetwork is recurrent and satisfies the locality and conservation conditions,

Then, an aggregated network B'=(P',T',I',O') with a set of transition rates R' is obtained by substituting the subnetwork N by one place pa, such that,

$$P' = \{P-P1\} \cup \{pa\} \ , \ T' = \{T-T1\},$$

$$I'(pi,tj) = I(pi,tj), \ O'(tj,pi) = O(tj,pi)$$

$$\forall \ pi \in P-P1 \text{ and } \ tj \in T-T1, \text{ and}$$

$$I'(pa,ti) = I(pj,ti) \ , \ O'(ti,pa) = O(ti,pj)$$

$$\forall \ pj \in P1 \text{ and } \ ti \in T-T1.$$

And if ri is the rate of an output transition ti $\notin$ Tout of the subnetwork ,i.e., I(pk,ti)=1 for some pk $\notin$ Pl, then the rate of such a transition in the aggregated network becomes marking dependent and is given by,

r'i(j) = ri . Pi(j) , where j is the number of tokens in pa at the current marking, and

Pi(j) = Pr[of finding at least one token in place pk of the subnetwork/given j].

These probabilities are obtained by solving the subnetwork N in isolation for the steady state probabilities for each possible value of j which defines the initial marking for N. The rates of transitions in {T-Tl-Tout} remain unchanged, i.e., for any ti $\notin$ {T-Tl-Tout}, ri' = ri.

Proof: We prove the above theorem, using the theory described in the previous Chapter, by showing that the above aggregation is actually a state aggregation of the process that describes the stochastic behaviour of the GSPN.

Assuming for simplicity that there exist no other immediate transitions in the GSPN (other than the ones in N), then the partition defined in definition 6 of the reachability set S into the subsets Sl and S2 is precisely the partition of S defined in equation (5.3.1) into ergodic classes at zero of the SDMP that describes the behaviour of the GSPN. Where Sl contains k markings, and S2 is further partitioned into l ergodic classes S2i,i=1,...,l (since N is a recurrent subnetwork). The matrices k' and K'' of equation (5.3.2) are

$$K' = \begin{array}{c} S21 \\ S22 \\ \cdot \\ \cdot \\ S21 \end{array} \begin{bmatrix} 1+ & & & \\ & 1+ & & \\ & & \cdot & \\ & & & 1+ \end{bmatrix}, \text{ and } K'' = \begin{array}{c} \begin{matrix} S21 & S22 & . . & S21 \end{matrix} \\ \begin{bmatrix} w_1^T & & & \\ & w_2^T & & \\ & & \cdot & \\ & & & w_1^T \end{bmatrix} \end{array}.$$

Where w_i as before is the steady state probability vector of a Markov chain with state space S2i. The trapping probability vectors do not appear here since there is no evanescent states. However, the subsets S2ij, j=1,...,r , that belong to the same equivalent class Sei as defined in definition 6 , will have the same probability vector w_i, which is obtained by solving the subnetwork N in isolation with an initial marking of i tokens in its input place.

The rate transition matrix A' of the aggregated process, given in equation (5.3.3), is now shown to be the same as the rate transition matrix of the aggregated network B'. Figure 6.2 shows a transition diagram between the various subsets of S. Where rini is the transition firing rate of a transition in Tin enabled by a marking in S1, r1 is the rate of a transition in {T-T1-Tin-Tout} enabled by all markings in S21, rin is the rate of a transition in Tin enabled by all markings in S21, and rout is the rate of a transition in Tout enabled by a marking in S21. The matrices A",B",C",and D" in equation (5.3.3) are,

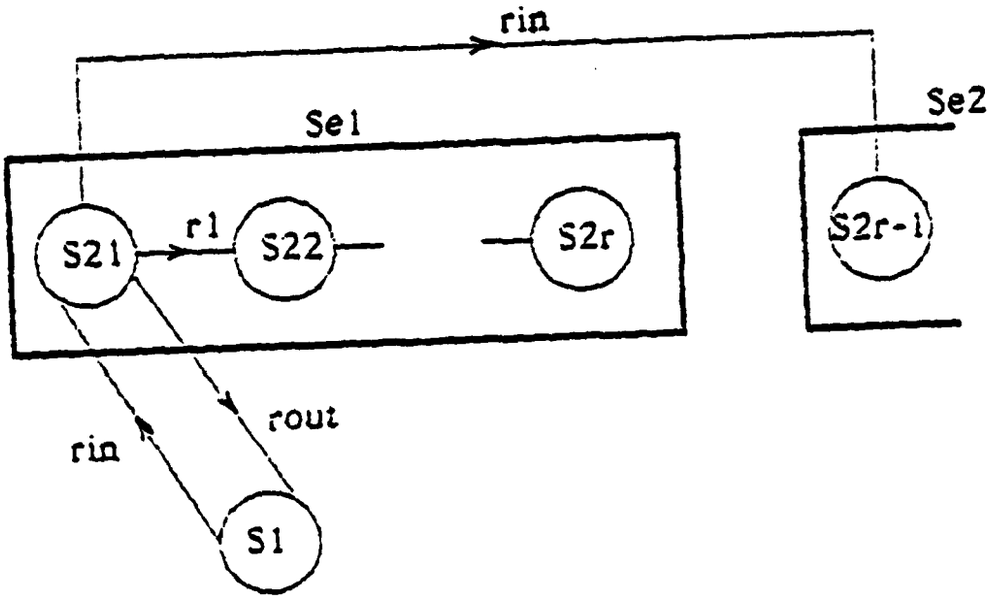


Figure 6.2

$$A'' = \begin{bmatrix} -r_{ini} \end{bmatrix} \qquad B'' = \begin{bmatrix} r_{ini} & & \\ & & \end{bmatrix}$$

$$C'' = \begin{bmatrix} r_{out} \\ \\ \end{bmatrix} \qquad D'' = \begin{array}{c} \\ S21 \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{ccc} S21 & & S22 \end{array} \quad S2r+1 \\ \begin{bmatrix} -(r_1+r_{in}) & r_1 & r_{in} \\ -(r_1+r_{in}) & r_1 & r_{in} \\ \vdots & \vdots & \vdots \\ -(r_1+r_{in}+r_{out}) & r_1 & r_{in} \\ & & \end{bmatrix} \end{array}$$

Then, the kx(s-1) martix B"K' (where s=k+1) will have the same nonzero elememts in B". And the matrices K"C" and K"D"K' are given by,

$$K''C'' = \begin{array}{c} i \\ \begin{matrix} 1 \\ 2 \\ \vdots \\ \vdots \\ 1 \end{matrix} \begin{bmatrix} r_{out}.w_{1i} \\ \\ \\ \\ \end{bmatrix} \end{array}, \quad K''D''K' = \begin{array}{c} 1 \qquad\qquad 2 \qquad r+1 \\ \begin{bmatrix} -(r_{out}.w_{1i}+r_1+r_{in}) & r & r_{in} \\ & & \end{bmatrix} \end{array}$$

Where w_{1i} is the probability of marking i in S21 that enables rout.

Clearly the only elements affected by the aggregation are the rates of the output transitions (transitions in To). In the general case, however, when there is more than one marking in an ergodic class that enables an output transition, the transition rate is multiplied by the sum of the probabilities of such markings. Which can be expressed as the probability of finding at least one token in the input place of such output transition. #

The above theorem defines an aggregation operation on a GSPN. The importance of this theorem for the approximate aggregation of SPNs will be demonstrated in the next Chapter.

The reduction operation described in the begining of this chapter can be generalized for a class of reducible subnetworks defined as follows,

Definition 11: A subnetwork N, defined in def. 4, is said to be reducible if, for each pi $\in$ Pout, there exist no transition tj $\in$ T1, such that I1(pi,tj) = 1. Therefore, there exist markings in each S2i, i=1,..l, that only enable transitions in Tout.

The reduction operation of a reducible subnetwork is given in the following proposition.

Proposition 1: for a live and bounded restricted GSPN B = (P,T,I,O), with an initial marking M1, reachability set S, and a set of transition firing rates R, if there exist a reducible subnetwork N = (P1,T1,I1,O1) as defined above, such that,

i) the set of input-output transitions Tin U Tout consists of timed transitions, and for each ti $\in$ Tin (ti $\in$ Tout), there exists only one place Pj $\in$ Pin (pj $\in$ Pout) such that,

O(ti,pj) = 1 (I(pj,ti) = 1), and

ii) N satisfies the locality and conservation conditions.

Then, a reduced network B' = (P',T',I',O') is obtained by replacing N, except for its places in Pout, by a set of timed transitions Ta such that,

P' = {P-P1} U Pout, T' = {T-T1} U Ta,

for all tj $\in$ {T-T1},

$I'(pi,tj) = I(pi,tj)$, $O'(tj,pi) = O(tj,pi)$, $\forall$ pi $\in$ {P-P1},

$I'(pi,tj) = I(pi,tj)$, $\forall$ pi $\in$ Pout,

for each ti $\in$ Tin, and for Pout = {po1,po2,..,pox}, define new transitions ti1,ti2,..,ti(x-1) $\in$ Ta such that

$O'(ti,po1) = 1$, $O'(ti1,po2) = 1$, $O'(ti2,po3) = 1$,.....,and

$O'(ti(x-1),pox) = 1$. Also

$O'(tis,pr) = O(ti,pr)$, $I'(pr,tis) = I(pr,tis)$, pr $\in$ {p-P1},

s = 1,...,(x-1).

(each ti $\in$ Tin is connected to the first place in Pout, and for each ti, (x-1) transitions, tis, s=1,...,(x-1), are defined in Ta that have the same input and output places in {P-P1} as ti. Each tis also has po(s+1) $\in$ Pout as an output place. Ta is the set of all transitions tis, s=1,...,(x-1), defined for each ti).

Also the set R' is defined as follows,

for each timed transition tk $\in$ {T-T1-Tin}, r'k = rk, and

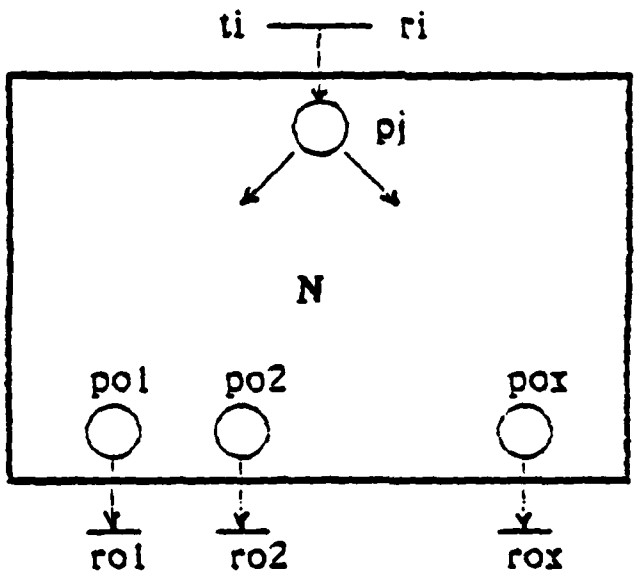
for each ti $\in$ Tin and its corresponding tis, s=1,...,x-1, in Ta,

$r'i = ri \cdot Pi(1)$, $r'is = ri \cdot Pi(s+1)$ for s = 1,....,x-1,

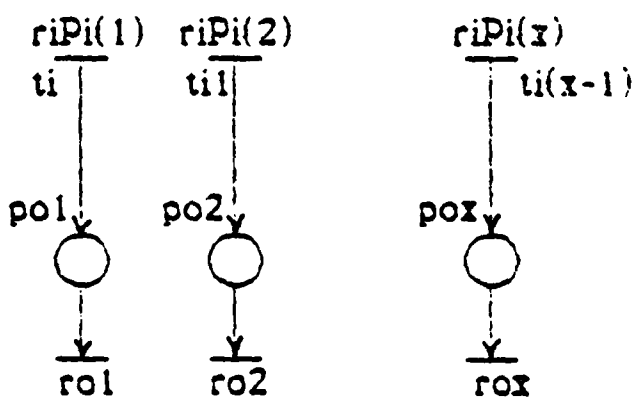
where Pi(j), j = 1,...,x, are the trappping probabilities of a token in the output place of ti in the set Pin to each one of the places in Pout, respectively.

Proof: we prove the above proposition, using again the theory described in the previous chapter, by showing that the above reduction operation corresponds to neglecting evanescent states in the process that describe the behaviour of the GSPN.

In figure 6.3(a), let ti be a transition in Tin with rate ri. Place pj is a place in Pin such that O(ti,pj) = 1. The set of


ti
ri
pj
N
po1
po2
pox
ro1
ro2
rox

(a)


riPi(1)
riPi(2)
riPi(x)
ti
ti1
ti(x-1)
po1
po2
pox
ro1
ro2
rox

(b)

Figure 6.3

places {po1,po2,...pox) is the set of output places Pout. And roi, i=1,...,x, are the rates of output transitions in Tout. Assuming for simplicity that there exist no other immediate transitions in the GSPN (other than the ones in N). Let Mil be a marking that enables ti, SN = {Mj1,Mj2,...,Mjk} be the set of markings that enable transition in T1, and the set SO = {Mol,Mo2,..,Mom} be the set of marking with at least one token in in any one of the places in Pout. Considering for simplicity markings with only one token in any one of the places of figure 6.3 (a), and let Mj1 be the marking in SN with one token in pj. Then the blocks VN and UN of matrices U and V that correspond to the above sets of markings are given by,

$$VN = \begin{array}{c} \\ Mil \\ Mol \\ Mo2 \\ \vdots \\ Mox \\ Mj1 \\ Mj2 \\ \vdots \\ Mjk \end{array} \begin{array}{c} \begin{array}{cccc} Mil & Mol & Mo2 \; . \; . & Mox \end{array} \\ \left[\begin{array}{cccc} & & & \\ & & I & \\ & & & \end{array}\right] \\ \left[\begin{array}{cccc} \emptyset & Pi(1) & . \; . \; . & Pi(x) \\ \emptyset & P2(1) & . \; . \; . & P2(x) \\ & & & \\ \emptyset & Pk(1) & . \; . \; . & Pk(x) \end{array}\right] \end{array}$$

$$UN = \begin{array}{c} \\ Mil \\ Mol \\ Mo2 \\ \vdots \\ M\emptyset x \end{array} \begin{array}{c} \begin{array}{ccccc} Mil & Mol & Mo2 \; . \; . & Mox & Mj1 \; Mj2 \; . \; . \; Mik \end{array} \\ \left[\begin{array}{cccc|ccc} & & & & & & \\ & & & & & & \\ & I & & & & \emptyset & \\ & & & & & & \\ & & & & & & \end{array}\right] \end{array}$$

where Pi(s) is the trapping probability from Mj1, and Pm(s) is the trapping probability from Mj1, 1 = 2,..,k, to Mos, s = 1,...,x. Also the corresponding block AlN of Al is,

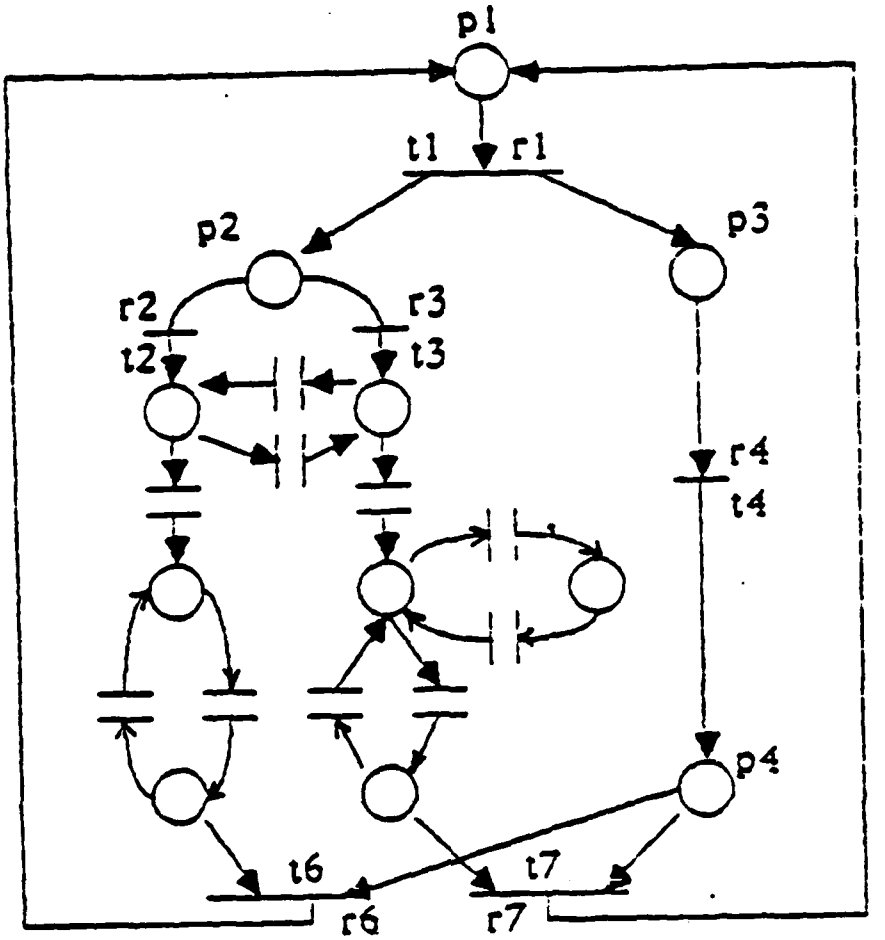
$$
A1N = \begin{array}{c} \\ Mil \\ Mol \\ \cdot \\ \cdot \\ Mox \\ Mjl \\ \cdot \\ \cdot \\ Mjk \end{array}
\begin{array}{c}
\begin{array}{cccccccc} Mil & Mol & . & . & Mox & Mjl & . & . & Mjk \end{array} \\
\left[\begin{array}{ccccc|ccc}
-ri & & & & & ri & & & \\
 & -ro1 & & & & & & & \\
 & & \cdot & & & & & & \\
 & & & \cdot & & & & & \\
 & & & & -rox & & & & \\
\hline
 & & & \emptyset & & & \emptyset & &
\end{array}\right]
\end{array}
$$

The corresponding block A'N of A' is, therefore, given by,

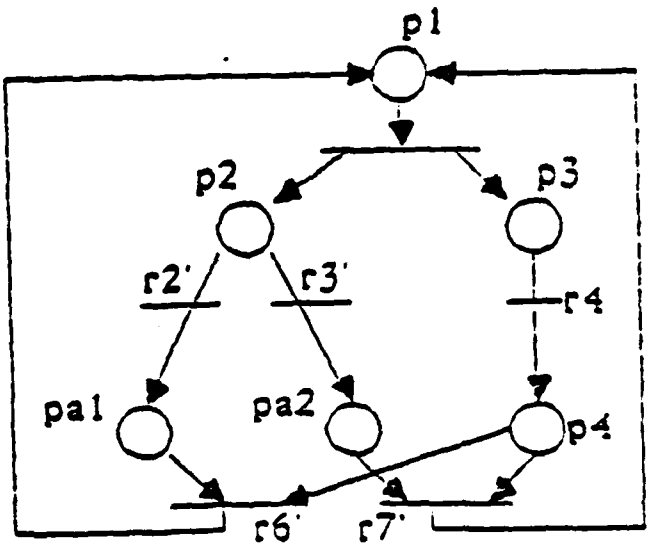
$$
A'N = \begin{array}{c} \\ Mil \\ Mol \\ Mo2 \\ \cdot \\ \cdot \\ Mox \end{array}
\begin{array}{c}
\begin{array}{ccccccc} -ri & ri.Pi(1) & ri.Pi(2) & . & . & ri.Pi(x) \end{array} \\
\left[\begin{array}{ccccccc}
 & -ro1 & & & & & \\
 & & -ro2 & & & & \\
 & & & \cdot & & & \\
 & & & & \cdot & & \\
 & & & & & -rox &
\end{array}\right]
\end{array}
$$

Clearly the the rate ri of the input transition is mutiplied by the trapping probabilities from Mjl only to markings that belong to SO. Figure 6.3(b) shows the reduction operation which produces the same matrix A'N. #

Example: Consider the GSPN in figure 6.4(a), the subnetwork of immediate transitions consists of reducible and recurrent parts. Using theorem 1, the recurrent part can be aggregated first to places pa1 and pa2. And the rates of transitions t6 and t7 are multiplied by the appropriate probabilities. Then using proposition 1, a reduction operation can be done on the remaining reducible subnetwork as shown in figure 6.4(b). Where Pj(k) is the trapping probability from a token in pij to pak, j,k=1,2.


p1
t1
r1
p2
p3
r2
r3
t2
t3
r4
t4
p4
t6
t7
r6
r7

(a)


p1
p2
p3
r2'
r3'
r4
pa1
pa2
p4
r6'
r7'

(b)

Figure 6.4

CHAPTER 7

APPROXIMATE AGGREGATION OF SPNs

7.1 Overview

In this chapter the analysis of SPNs by approximate aggragation and lumping is considered. In section 7.2, the approximate hierarchical aggregation of SPNs is demonstrated by several examples. And in section 7.3, the approximate lumping parallel transitions in a SPN is considered.

7.2 Hierarchical Aggregation of SPNs

The analysis of SPNs with transition rates of different orders of magnitude can be greatly simplified using approximate aggregation techniques. In this section, the analysis of such SPNs will be considered. And we demonstrate by several examples that the analysis of singularly perturbed SPNs can be reduced to the analysis of that of a hierarchical sequence of subnetworks, each of which is valid at a certain time scale. Since the time behaviour of a SPN is isomorphic to a continuous time MC, the hierarchical aggregation of SPNs is equivalent to that of MCs described in chapter 4. However, as was the case for queuing networks, such aggregation at the SPNs level is much more advantageous than the aggregation of the correspoding MC when the state space is very large. This is because at the SPN level we are dealing with the aggregation of subnetworks, whereas at the MC level we aggregate groups of large number of states.

The exact aggregation, defined in the previous chapter for subnetworks consisting of immediate transitions is a GSPN, can be employed for subnetworks consisting of fast transitions in an

SPN. However, the aggregation is approximate since fast transitions have very large, yet finite, firing rates compared to slow transitions. The following example illustrates the above concept.

Example 7.1: consider the SPN shown in figure 7.1, where r1,r2,r3, and r4 are large compared to r5 and r6. Considering large transitions only with input-output places, the SPN is decomposed into the recurrent subnetworks N1 anf N2 shown in figure 7.2(a). Using the theory developed in the previous chapter, these subnetworks can be aggregated, and an aggregated SPN with slow transitions can be obtained as shown in figure 7.2(b). Where r'5(i) and r'6(j) are state dependent rates given by,

r'5(i) = r5 P1(i) , and r'6(j) = r6 P2(j), where

P1(i) = Pr[of finding at least one token in place p2 of N1 when there are i tokens in place pa1], and

P2(j) = Pr[of finding at least one token in place p4 of N2 when there are j tokens in pa2].

The above probabilities are obtained by solving subnetworks N1 and N2 for all possible markings of the aggregated SPN, then

$P1(1) = r1/(r1+r2)$, $P2(1) = r3/(r3+r4)$,

$P1(2) = r1\ r2/(r1^2+r1r2+r2^2)$, $P2(2) = r3\ r4/(r3^2+r3r4+r4^2)$

The rate transition matrix A' of the aggregated SPN is given by,

$$A' = \begin{bmatrix} -r'5(2) & r'5(2) & 0 \\ r'6(1) & -(r'5(1)+r'6(1)) & r'5(1) \\ 0 & r'6(2) & -r'6(2) \end{bmatrix}$$

Which can be solved for the steady state probabilities of the

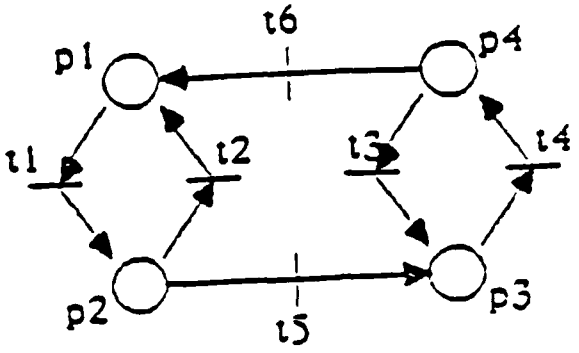
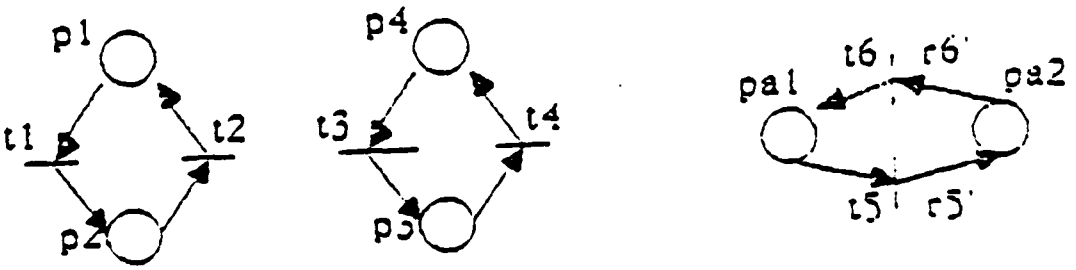


Figure 7.1



(a) (b)

Figure 7.2

markings of the aggregated SPN.

To show that the above is equivalent to the approximate aggregation of the isomorphic MC of the SPN in figure 7.1, the reachability set of this SPN and the correspoding rate transition matrix are given by,

	p1	p2	p3	p4
M1	1	0	1	0
M2	1	0	0	1
M3	0	1	1	0
M4	0	1	0	1
M5	2	0	0	0
M6	1	1	0	0
M7	0	2	0	0
M8	0	0	2	0
M9	0	0	1	1
M10	0	0	0	2

$$A = \begin{array}{c} \\ M1 \\ M2 \\ M3 \\ M4 \\ M5 \\ M6 \\ M7 \\ M8 \\ M9 \\ M10 \end{array} \begin{bmatrix} -x1 & r4 & r1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ r3 & -x2 & 0 & r1 & r6 & 0 & 0 & 0 & 0 & 0 \\ r2 & 0 & -x3 & r4 & 0 & 0 & 0 & r5 & 0 & 0 \\ 0 & r2 & r3 & -x4 & 0 & r6 & 0 & r5 & 0 & 0 \\ 0 & 0 & 0 & 0 & -r1 & r1 & 0 & 0 & 0 & 0 \\ r5 & 0 & 0 & 0 & r2 & -x6 & r1 & 0 & 0 & 0 \\ 0 & 0 & r5 & 0 & 0 & r2 & -x7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -x8 & r4 & 0 \\ r6 & 0 & 0 & 0 & 0 & 0 & 0 & r3 & -x9 & r4 \\ 0 & r6 & 0 & 0 & 0 & 0 & 0 & 0 & r3 & -x10 \end{bmatrix}$$

Where xi is the sum of the all elements in row i, i=1,..,10.

Since the r1,r2,r3,and r4 >> r5 and r6, the above matrix can be written as,

A = A(p) = AØ + p A1, where p = r5 + r6 , is the maximum degree of coupling between aggregates [COUR 77], and

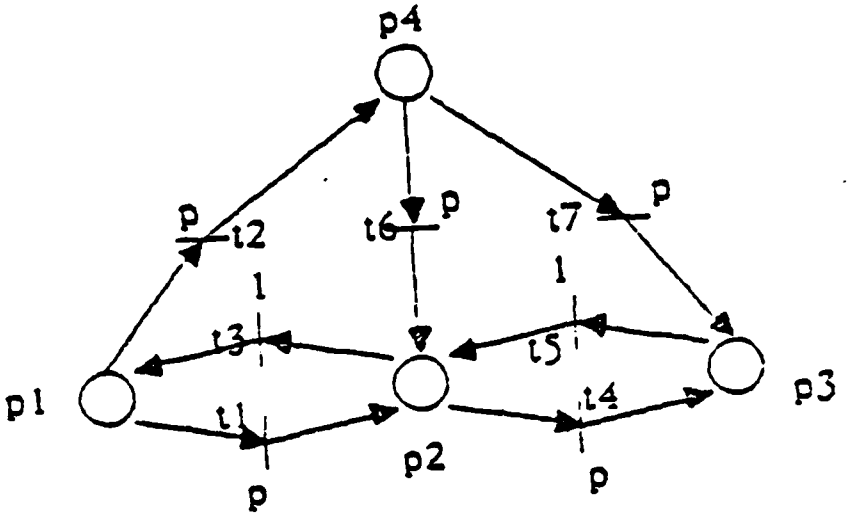
AØ = A(Ø), is obtained by letting r5 and r6 equal Ø in A. Then from theorem 1 of chapter 4, and since rank A(p) > rank AØ (i.e, the process is singularly perturbed), the steady state transition probability matrix Pr of the MC is given by,

$Pr = \lim_{t \to \infty} \exp\{A\ t\} = U \exp\{A''\ p\ t\}\ V$, where

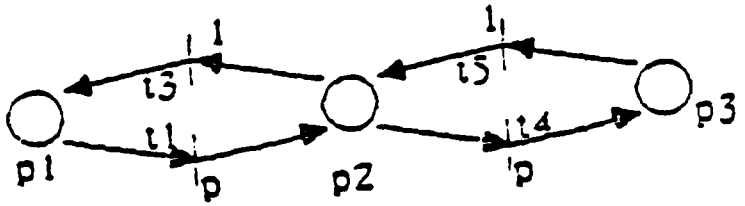
$Z = \lim_{t \to \infty} \exp\{A\emptyset\ t\} = V.U$ is the canonical decomposition of Z, and p A" = U p A1 V , is the rate transition matrix of the aggregated process. It can be easily shown that this matrix is the rate transition matrix A' of the aggregated SPN of figure 7.2(b).

The approximate aggregation of the SPN of figure 7.1 produces a hierarchical decomposition of the SPN at two time scales. At the fast time scale t the SPN reduces to subnetworks N1 and N2 of figure 7.2(a). And at the slow time scale t/p, the aggregated SPN of figure 7.2(b) is obtained. The current marking of the aggregated SPN at the higher level of the hierarchy determines the number of tokens in the subnetworks at the lower level. Which are solved to determine the rates of transitions at the higher level. Such hierarchical decomposition is symptomatic of singularly perturbed SPNs defined as follows,

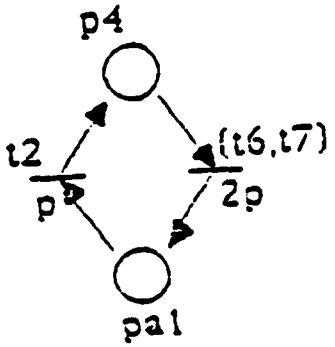
Definition 1: a SPN with fast and slow transitions is said to be singularly perturbed, if and only if the corresponding MC is singularly perturbed, i.e., if rank A > rank AØ, where A is the rate transition matrix and AØ is the matrix A when all slow



(a)



(b)



(c)

Figure 7.3

transitions are set to zero.

In the rest of this chapter the analysis wil be focused on irreducible SPNs defined as follows,

Definition 2: a SPN = (P,T,I,O), with an initial marking M1 and a reachability set S, is said to be irreducible if it is live and recurrent, i.e., if for any ti ϵ T and for all Mj ϵ S, there exists a transition firing sequence from Mj in which ti fires. And if for any Mj,Mk ϵ S, Mk is reachable from Mj.

The recurrence of markings and the liveness issues are related, however they are not equivalent. It is clear that a recurrent PN is not necessarily live, and not all live and bounded PNs are recurrent. Molloy [MOL81] proved that any live and bounded PN, with a reachability set S, has a unique subset of recurrent markings S' $\subset$ S. Which entirely describes the steady state behaviour of the SPN. In the above definition of an irreducible SPN S' = S.

The following proposition characterizes singularly perturbed SPNs.

Proposition 7.1: an irreducible SPN with fast and slow transitions, and a reachability set S, is singularly perturbed if and only if one of the following conditions is satisfied,

i) There exists more than one marking in S which enable only slow transitions,

ii) There exist at least two disjoint recurrent subnetworks consisiting of fast transitions, or

iii) There exist at least one marking as defined in i), and one subnetwork as defined in ii).

Proof: from definition 1, a SPN is singularly perturbed if and only if, in its corresponding MC rank A > rank A∅, or equivalently nul.A < nul.A∅. Since for an irreducible SPN nul.A=1, then

a) if condition i) is satisfied, then clearly nul.A∅ > 1 (there exist more than one row of zeros in A∅).

b) if condition ii) is satisfied, then each subnetwork will produce at least one block diagonal matrix in A∅, the nulity of which will be equal to 1. And therefore nul.A∅ > 1.

c) clearly from the above if condition iii) is satisfied, then again nul.A∅ > 1.

We prove the necessity of the above conditions by contradiction as follows. Suppose that none of the above conditions are satisfied, yet nul.A∅ > 1. Then considering only fast transitions, there exist more than one ergodic classes of markings Ei,i=1,2,..,. Let E be the set of all ergodic classes. if any Ei $\in$ E contains more than one marking, then these markings must be reachable from each other by fast transitions (recurrent). But since condition ii) is not satisfied, then there exist at most one Ei $\in$ E with more than one marking. And each of the remaining classes consists of a single marking. Now, since conditions ii) and iii) are not satisfied, then these single markings do not enable any slow transition. And no other transition in the SPN is enabled. Then nul.A > 1, i.e. the SPN is is not live which is a contradiction. #

As mentioned before, the hierarchical decomposition of at different time scales of perturbed SPNs is symptomatic of

singular perturbation. The analysis of regularly perturbed SPNs reduces approximately to the analysis of the network under fast transitions only. We illustratees the above concepts by the following examples.

Example 7.2: Consider the SPN shown in figure 7.3(a), where the rates of slow transitions are modeled by a small parameter $p<<1$. The reachability set is given by

	p1	p2	p3	p4
M1	1	0	0	0
M2	0	1	0	0
M3	0	0	1	0
M4	0	0	0	1

From proposition 7.1, this subnetwork is singularly perturbed since marking M1 amd M4 enable only slow transitions. At the fast time scale, the SPN reduces to the subnetwork consisting of transitions t3 and t5, and their set of input output places P1 = {p1,p2,p3}. However, as shown in figure 7.3(b), slow transitions t1 and t4, with input output places in P1, are also included in the subnetwork. Although these slow transitions can be neglected at the fast time scale, their inclusion here is for improving the acurracy of the approximation. This subnetwork is now recurrent and can be aggregated into one place pa1. Figure 7.3(b) shows the aggregated SPN at the slow time scale consisting of slow transitions only. This SPN can be solved for the probabilities of the markings M1' and M2' defined as follows,

	pa1	p3
M1'	1	0
M2'	0	1

Thus P(M1') = 2/3 , and P(M2') = 1/3 , and from the subnetwork in figure 7.3(b),

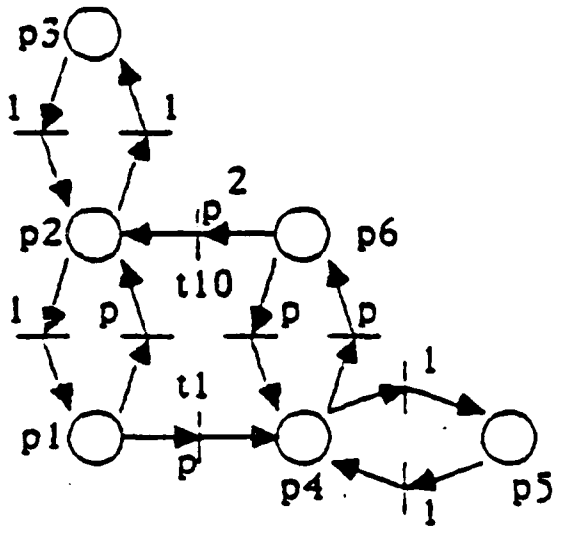
$P(M1) = P(M1') \cdot 1/(1+p+p^2)$, $P(M2) = P(M1') \cdot p/(1+p+p^2)$,

$P(M3) = P(M1') \cdot p^2/(1+p+p^2)$, and P(M4) = P(M4')

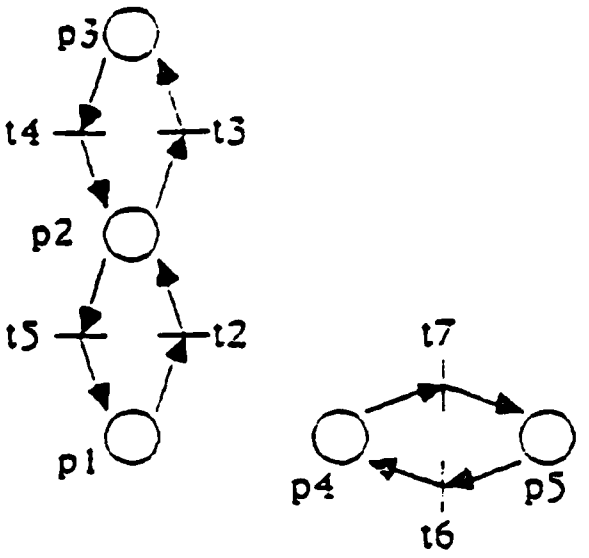
It can be easily seen that without the inclusion of slow tramsitions t1 and t4 in the fast time scale,the probabilities P(M2) and P(M3) would be zeros.

Example 7.3: Consider the SPN shown in figure 7.4(a), where $p \ll 1$. Again this SPN is singularly perturbed, since there exist two recurrent subnetworks N1 and N2 with fast transitions. Where N1 = {p3,p2,t3,t4} and N2 = {p4,p5,t6,t7}. At the fast time scale t, the SPN is decomposed into the subnetworks shown in figure 7.4(b). Where again the slow transition t5 is included since its input and output places p2 and p1 belong to the set of input output places of fast transitions in one of the subnetworks in figure 7.4(b). The inclusion of this transition in this example is crucial since the probability of finding a token in p1 would be zero otherwise. And since p1 is an output place of this subnetwork to the rest of the network, the rate of the output transition t1 depends on the above probability.

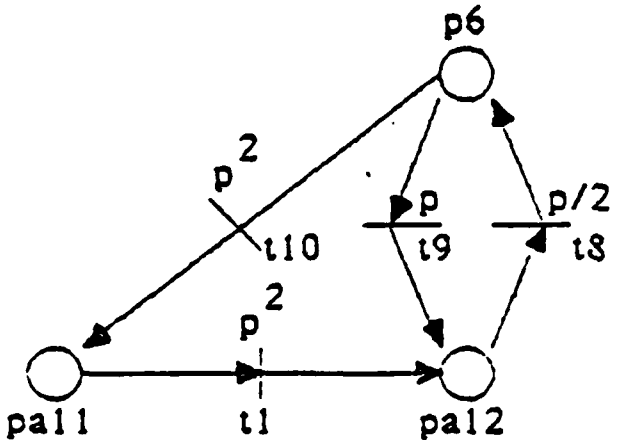
At the slow time scale t/p the aggregated SPN shown in figure 7.4(c) is obtained. Where the subnetworks in figure 7.4(b) are aggregated into places pa11 and pa12. This SPN is also singularly perturbed. Since it consists of slow and fast transitions (p and



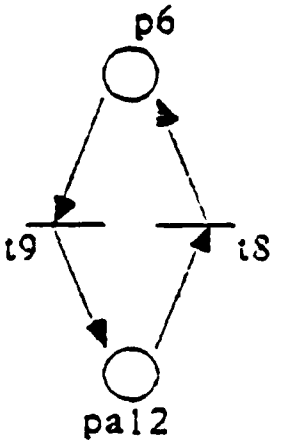
(a)



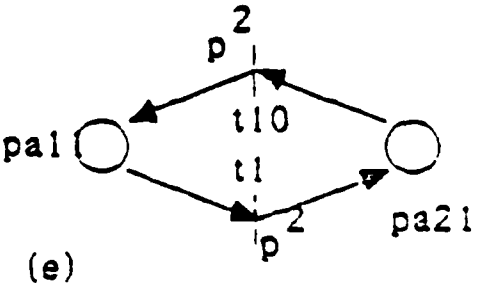
(b)



(c)



(d)



(e)

Figure 7.4

p^2). And there exists a recurrent subnetwork of fast transitions {pa12,p6,t8,t9}, and a marking that only enables the slow transition t1. Therefore at time scale t/p it reduces to the subnetwork of fast transitions shown in figure 7.4(d). and at time scale t/p^2, the aggregated SPN of slow transition shown in figure 7.4(e) is obtained. Therefore, the networks in figures 7.4(b,d,e) are the time scale decomposition of of the SPN in figure 7.4(a) at t, t/p, and t/p^2 respectively.

7.2 Approximate Lumping :

In this section, another method that reduces the analysis complexity of SPNs will be discussed. Since SPNs are isomorphic to MCs, state lumping defined for MCs can be implemented on subnetworks of SPNS. We first review the notion of "lumpability" [KEM 67, COU 77, DEL 84] of an irreducible finite state Markov process (FSMP), and demonstrate by an example the application of this notion to SPNs.

Let x_t be a discrete parameter homogenous MC, with state space E={1,2,...,n}, and a transition probability matrix P. Let $\{Q_1,Q_2,....,Q_R\}$ be a partition of the set E. Each subset Q_i; i=1,2,....,R can be considered as a state of a new process. Let s_t denote the state occupied by this new process at time t. The probability of a transition occuring at time t from state Q_i to state Q_j, $P'_{QiQj}(t)$, is given by,

$$P'_{QiQj} = Pr\{ s_t=Q_j / s_{t-1}=Q_i, s_{t-2}=Q_k,...,s_0=Q_m\}.$$

The original MC is thus reduced to a stochastic process with fewer states. The new process is called a lumped process, and Q_i a lumped state.

The lumped process is again a homogenous MC only if it is time independant and depends only on s_{t-1}, i.e.

$$P'_{Q_iQ_j}(t) = Pr\{ s_t=Q_j / s_{t-1}=Q_i\}$$
$$= P_{Q_iQ_j} \qquad t>0 \qquad (7.2.1)$$

Kemeny and Snell [KEM 67] qualify x_t as being lumpable if equation (7.2.1) is true for every possible initial state. Defining

$$P_{iQ_j} = \sum_{j \in Q_j} P_{ij}$$

as the probability of moving from state i to set Q_j, then the MC is lumpable with respect to the partition $\{Q_1,Q_2,....,Q_R\}$ if and only if all the P_{iQ_j} have the same value for every $i \in Q_i$, and for any given $Q_j \neq Q_i$.

Similarly, for a continuous time, homogeneous FSMP $\{x(t),t\geq 0\}$, with a state space E, and a rate transition matrix A. Then x(t) is lumpable with respect to a partition $\{Q_1,Q_2,....,Q_R\}$, if and only if

$$AU = UA' \qquad (7.2.2)$$

where U is (n,R) partition matrix, whose (i,Q_j) entry is

$$U_{iQ_j} = 1 \qquad \text{if } i \in Q_j$$
$$= 0 \qquad \text{otherwise}$$

$$, \forall\, i \in E,\ j = 1,2,....,R$$

A' is the rate transition matrix of the lumped process $\{x'(t), t\geq 0\}$. Equation (7.2.2) can be written as

$$\sum_{k \in Q_k} a_{ik} = a'_{Q_jQ_k} \ , \forall\, i \in Q_j,\ k=1,2,....R.$$

That is the rate of going from $j \in Q_j$ to group Q_k depends only on Q_j and is independant of i.

As an example,consider the MC shown in figure 7.5(a). If $a_{35} = a_{45} = a$, then a lumped MC can be obtained as shown in

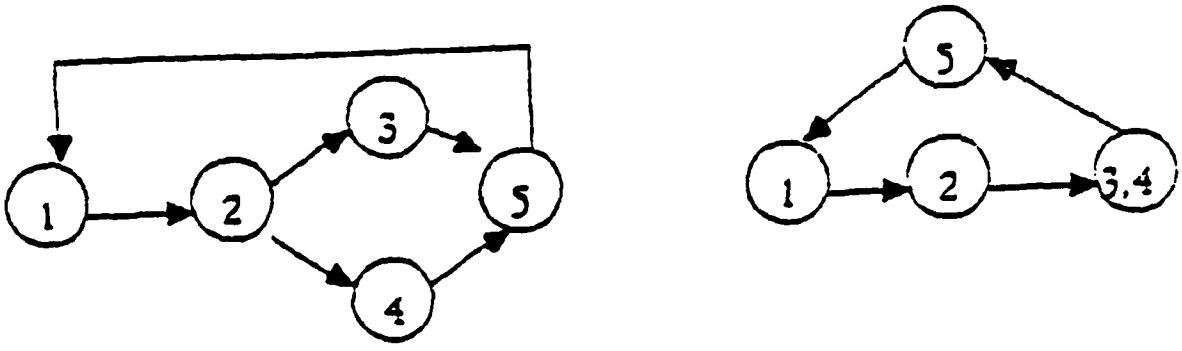
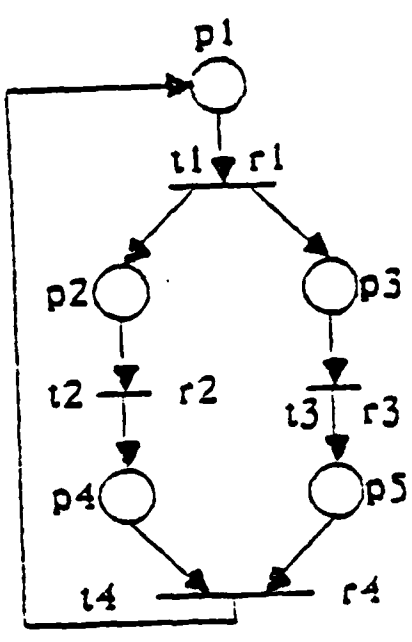


Figure 7.5



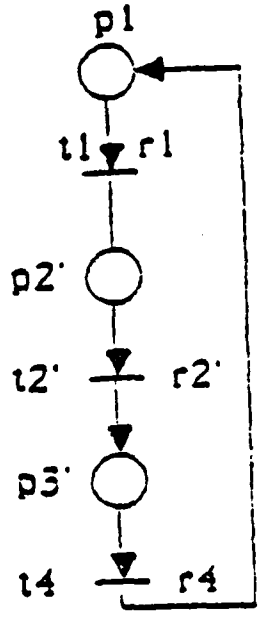


Figure 7.6

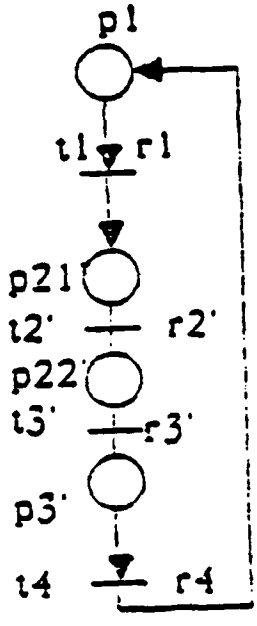


Figure 7.7

figure 7.5(b), where

$a_{2,\{3,4\}} = a_{2,3} + a_{2,4}$, and

$a_{\{3,4\},5} = a$

We demonstrate the application of the above concept to SPNs by the following example.

Consider the SPN shown in figure 7.6(a), the reachability set of which is given by,

M_1 1 0 0 0 0

M_2 0 1 1 0 0

M_3 0 1 0 0 1

M_4 0 0 1 1 0

M_5 0 0 0 1 1

Let us investigate the probability of lumping the subnetwork $\{p_2,p_3,t_2,t_3\}$ consisting of the parallel transition t_2 and t_3 together with their input output places, to obtain the lumped SPN shown in figure7.6(b), the reachability set of which is

M'_1 1 0 0

M'_2 0 1 0

M'_3 0 0 1

Clearly, we are investigating the lumping of marking M_2, M_3, and M_4 into M'_2. Let x be a random variable representing the firing time of transition t'_2. It can be easily seen that

$x = \max(x_1,x_2)$, where x_1 and x_2 are r.v.s representing the firing times of transitions t_1 and t_2 respectively. Since x_1 and x_2 are independant and exponentially distributed withrates r_1 and

r_2 , then,

$$P\{x \leq t\} = P\{x_1 \leq t\} . P\{x_2 \leq t\}$$
$$= (1-e^{-r2t}) . (1-e^{-r3t})$$

Therefore, even if $r_1 = r_2$, x is not exponentially distributed, and since by the definition of a SPN, every transition must be associated with an exponentially distributed firing time, the lumped network in figure 7.6(b) is not defined. However, we can approximate the distribution of x by an exponential distribution with rate r, such that the folloing integral is minimized,

$$\text{Min}_r \int_0^\infty [(1-e^{-r2t})(1-e^{-r3t}) - (1-e^{-rt})]^2 \, dt$$
$$, r, r2, r3 > 0$$

Then $2r^2(r+r_3)^2(r_2+r_3+r)^2 + 2r^2(r+r_2)^2[2r_2(r+r_3)+r_2^2] -$
$$1/2(r+r_2)^2(r+r_3)^2(r+r_2+r_3)^2 = 0$$

which can be solved iteratively for r.

Let $r_1 = r_2 = r_3 = 1$, $r_4 = \infty$ (i.e. t_4 is an immediate transition)

Then, from the above, r = 0.649. Solving the lumped SPN for the steady state probabilities of its marking, we get

$$P(M'_1) = 0.394 , P(M'_2) = 0.606 , (P(M'_3) = 0)$$

And solving the original SPN ,

$$P(M_1) = 0.4 , P(M_2)+P(M_3)+P(M_4) = 0.6$$

For $r_1 = 1$, $r_2 = 2$, $r_3 = 3$, $r_4 = 4$, then r = 1.546 and

$P(M'_1) = 0.5271$, $P(M'_2) = 0.341$, $P(M'_3) = 0.1317$, $P(M_1) = .5309$, $P(M_2)+P(M_3)+P(M_4) = 0.336$, $P(M_5) = 0.1327$, error ~ 0.7%

If the transition rates are state (Marking) dependant, then the above will not hold, since x_1 and x_2 are no longer independant. For this case, let us consider the lumped SPN shown

in figure 7.7 ,where the subnetwork of parallel transitions t_2 and t_3 are lumped into a subnetwork consisting of the two serial transitions t'_2 and t'_3. The reachability set is given by,

M'_1 1 0 0 0

M'_2 0 1 0 0

M'_3 0 0 1 0

M'_4 0 0 0 1

In this case, marking M_3 and M_4 were lumped into M'_3. Therefore,

$r'_2 = r_2(2) + r_3(2)$

where $r_i(2)$;i=2,3 , are the transition rates of t_2 and t_3 at marking M_2.

Now let x be a r.v. representing the firing time of transition t'_3, then

$$P(x< t) = P(x_1<t)r_3(2)/[r_2(2)+r_3(2)] + P(x_2< t)r_2(2)/[r_2(2)+r_3(2)]$$

$$= 1 - r_3(2)e^{-r2(3)t}/[r_2(2)+r_3(2)] - r_2(2)e^{-r3(4)t}/[r_2(2)+r_3(2)]$$

which reduces to the exponential distribution in two cases ; case 1 when $r_2=r_3$ and case 2 when either r_2 or r_3 tends to infinity. Define r'_3 as the mean of the above distribution, then,

$$r'_3 = r_3(2)/(r_2(3))(r_2(2)+r_3(2)) + r_2(2)/(r_3(4))(r_2(2)+r_3(2))$$

Approximating the above distribution with an exponential distribution with rate r'_3, then the lumped SPN can be solved for the steady probability distribution.

Let $r_1 = 1$, $r_2(2) = 3$, $r_2(3) = 6$, $r_3(2) = 2$, $r_3(4) = 4$, $r_4(4) = \infty$

Then $r'_2 = 5$ and $r'_3 = 4.615$

Then $P(M'_1) = 0.7058$, $P(M'_2) = 0.14117$, $P(M'_3) = 0.15295$

The exact solutionis,

$P(M_1)=0.7058$, $P(M_2)=0.14116$, $P(M_3)+P(M_4)=0.15285$

and for $r_1=1$, $r_2(2)=4$, $r_2(3)=6$, $r_3(2)=6$, $r_3(4)=10$, $r_4=$.

Then $r'_2=10$, $r'_3=7.1428$

$P(M'_1)=0.8076$, $P(M'_2)=0.08076$, $P(M'_3)=0.11154$

and the exact solution is

$P(M_1)=0.8064$, $P(M_2)=.08064$, $P(M_3)+P(M_4)=.1128$ ~ 1.1% error.

CHAPTER 8

OTHER APPLICATIONS

8.1 Overview

In this chapter two other applications of GSPNs as an analytical modeling tool will be considered. In section 2, analytical models for systems Reliability and Maintainablity (R&M) are considered. In section 3, a model for systems fault diagnosis is considered.

8.2 Modeling Systems Reliability and Maintainability

The development of cost-effective analytical models for complex systems R & M has been an active area of research [DOL 83, FLE 84a, 84b, 85]. Generally speaking, the model can be defined on the basis of the following information : system structure, maintenance description, module or component data. A complex system can often be divided into subsystems and modules. A module is a replicable subset of a system chosen to be modeled as a unit, on which system design, input data and maintenance policies are based. Subsystems are sets of dependant modules which are redundant, or with interdependant maintenance strategy. Subsystems are usually defined to be independant and connected in series. Thus a failure of a subsystem would cause system failure. Maintenance description includes facilities, space modules, test equipment, and maintenance policies such as inspection, replacements, and repair procedures. Module data includes failure rates, repair rates, fraction of faults detected, fraction of faults isolated, and false alarm rate.

The authorsin [FLE 84b] appeared to be the first ones to develop a MC model which charactarizes all key elements insystem R & M analysis. The state of the MC chain model is defined by the number of operational, or failed modules and the maintenance actions in progress. Even though a MC can be constructed for each subsystem, the model becomes intractable for complex systems due to state space explosion. In this section we consider the use of GSPNs for modeling systems R & M. GSPNs allow the activities to be modeled at a high level of abstraction. In other words, GSPNs provide a more detailed, yet simpler, description of system activities. By detecting activities with duration of different orders of magnitude, approximate aggregated models can de obtained. This aggregation further reduces the complexity of the analysis. The above concept is demonstrated by the following example.

Example 8.1: Consider a system which consists of several redundant modules. A built-in testing (BIT) unit monitors the operational status of all modules. A module that fails and its failure is covered , i.e. detected and isolated, by the BIT, is removed from the system and sent to the shop for repair. The removal is considered as a maintenance action at the organizational ("O") level [FEL 84b]. A failed module that was not immediatly detected will produce an error after some time. The system will then be inspected and the failed module will be removed and sent to the shop for repair. The inspection and the removal of the module is considered as another maintenance action at the "O" level. A fault that was detected but could not be isolated causes a system

failure. In this case, all modules are sent to the shop for repair. The input parameters needed by the model are summarized as follows :

m_i = No. of modules	2
Mu = module failure rate	10^{-5} failures/hr.
FA = false alarm rate	varies
r_1 = mean time for "O" level inspect and remove	2 hrs.
r_2 = mean time for "O" level remove	1 hr.
r_3 = mean time for repair	72 hrs.
d = fraction of faults detected	varies
i = fraction of faults isolated	varies
FD = mean time to detect an immediately undetected fault	100 hrs.

The above parameters are taken from an example given in [FEL 84b] for a mission computer in an air craft radar set. The false alarm rate is defined as the rate at which the BIT detects a fault while the system is fault-free. Operational data of some systems show that often 50% of all maintenance actions are due to false alarms. Figure 8.1 (a) shows a GSPN which models system R & M of the above example. The initial marking of the GSPN indicates that there are two modules installed and active, no maintenance, and the system is up. An important advantage of using the GSPN is that the graphical complexity does not change with the increase of the number of modules in the system. Note that the component failure rate and false alarm rate are much smaller than detection and repair rates. An aproximate aggregated GSPN can thus be obtained (figure 8.1(b)). In figure 8.1 (b), r_1 = (FA+2Mu d)i, r_2=r_1(i-1)/i, and r_3= 2Mu(1-d). The basic assumption in the

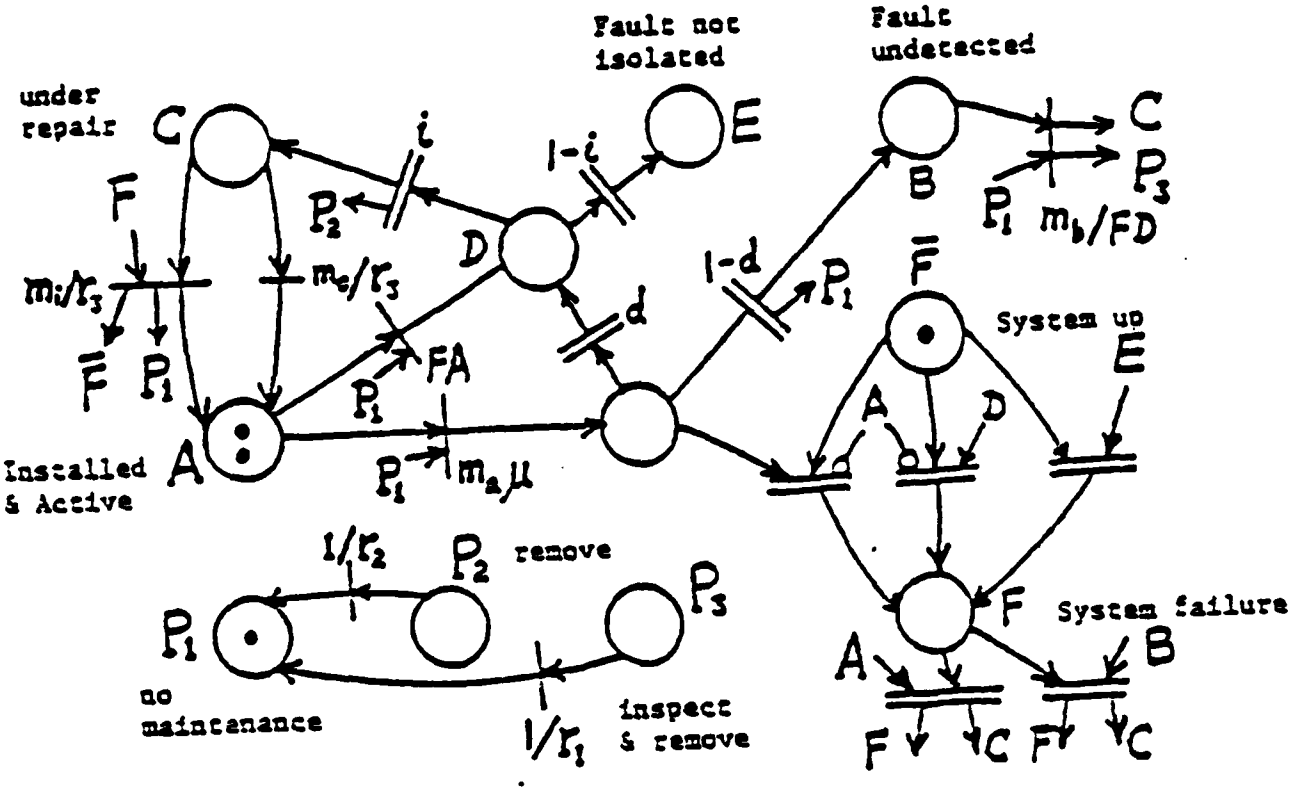


Figure 8.1 (a)

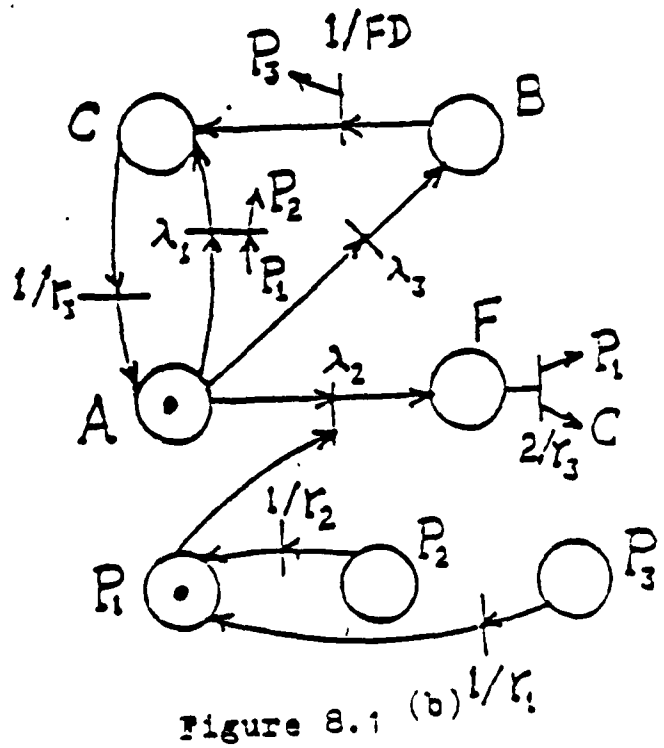


Figure 8.1 (b)

aggregated GSPN is that a false alarm of a second failure is not posible with one failure. Such a model can then be analyzed to obtain system unavailability, which is the probability of finding a token in F, and the average number of maintenance action per million hours. The approximation was found to be accurate for imperfect isolation (i<1).

Figure 8.2 shows the impact of false alarm on system unavailability for differnt values of detection and isolation factors. The false alarm is increased up to a value where 50% of all maintenance actions are due to false alarms. It demonstrates that the isolation factor plays a key role in the impact of false alarm on unavailability. The higher the isolation factor is, the lower is the impact of false alarm on unavailability. This is because an alarm which could not be isolated causes a system failure. The detection factor, on the other hand, does not seem to have significant effect on the impact of false alarms. However, the lower the value of d in a system with false alarms, the lower is the unavailability. This is because an error caused by an undetected fault requires an inspection which is assumed by the model to detect any previously undetected faults. The above phenomena can be seen more clearly from the GSPN model of figure 8.1(a). This also demonstratesthat the GSPN model can facilitate the study of system activities.

8.3 Modeling Fault Diagnosis

In this section, a model for fault diagnosis (fault isolation) will be presented using SPNs. The model is defined by

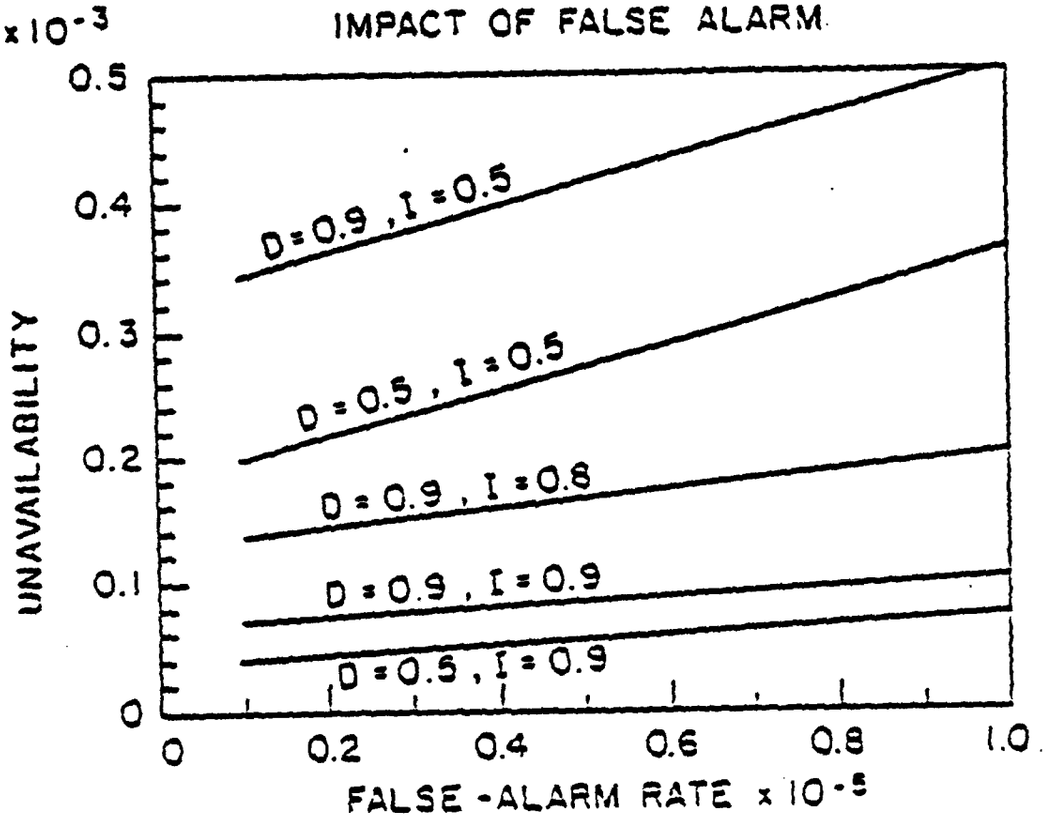

IMPACT OF FALSE ALARM
x10⁻³
UNAVAILABILITY
0.5
0.4
0.3
0.2
0.1
0
D=0.9 , I=0.5
D=0.5 , I=0.5
D=0.9 , I=0.8
D=0.9 , I=0.9
D=0.5 , I=0.9
0
0.2
0.4
0.6
0.8
1.0
FALSE-ALARM RATE x10⁻⁵

Figure 8.2

topological and functional descriptions of the system (the number of modules, connectivity description, and module functions), relative a priori module failure probabilities [PIP 84], and the specified normal range for each module I/O value. The concept is described trough the following example.

Example 8.2: Consider a system which consists of three modules as shown in Figure 8.3(a). The function of module 1 is such that an abnormal input will still yield a normal output. Modules 2 and 3 are linear, i.e. a bad input causes a bad output. Figure 8.3(b) shows a GSPN model for the fault diagnosis. For each input and output of a module, two places, say y_1 and y_2, are assigned. The subscript 1 stands for a normal value while 2 stands for an abnormal one.

Transitions between the input-output places of a module represent data flow activities. Probabilities assigned to conflicting transitions in each module are computed from the relative a priori module failure probabilities, which is assumed to be much less than one . A token in place 1, 2 or 3 indicates that the corresponding module is bad. Given the measurments $x = x_1$ and $z = z_2$, the GSPN can be solved to determine the probability that a certain module is faulty. As an example, suppose that a priori failure probabilities of all modules are equal to 0.1. The distributions oftokens in places 1, 2 and 3 are as follows: 100 with prob. 0.09, 101 with prob. 0.01, 010 with prob. 0.81, and 011 with prob. 0.09. Therefore, the above measurements lead to the conclusion that module 2 is most likely faulty.

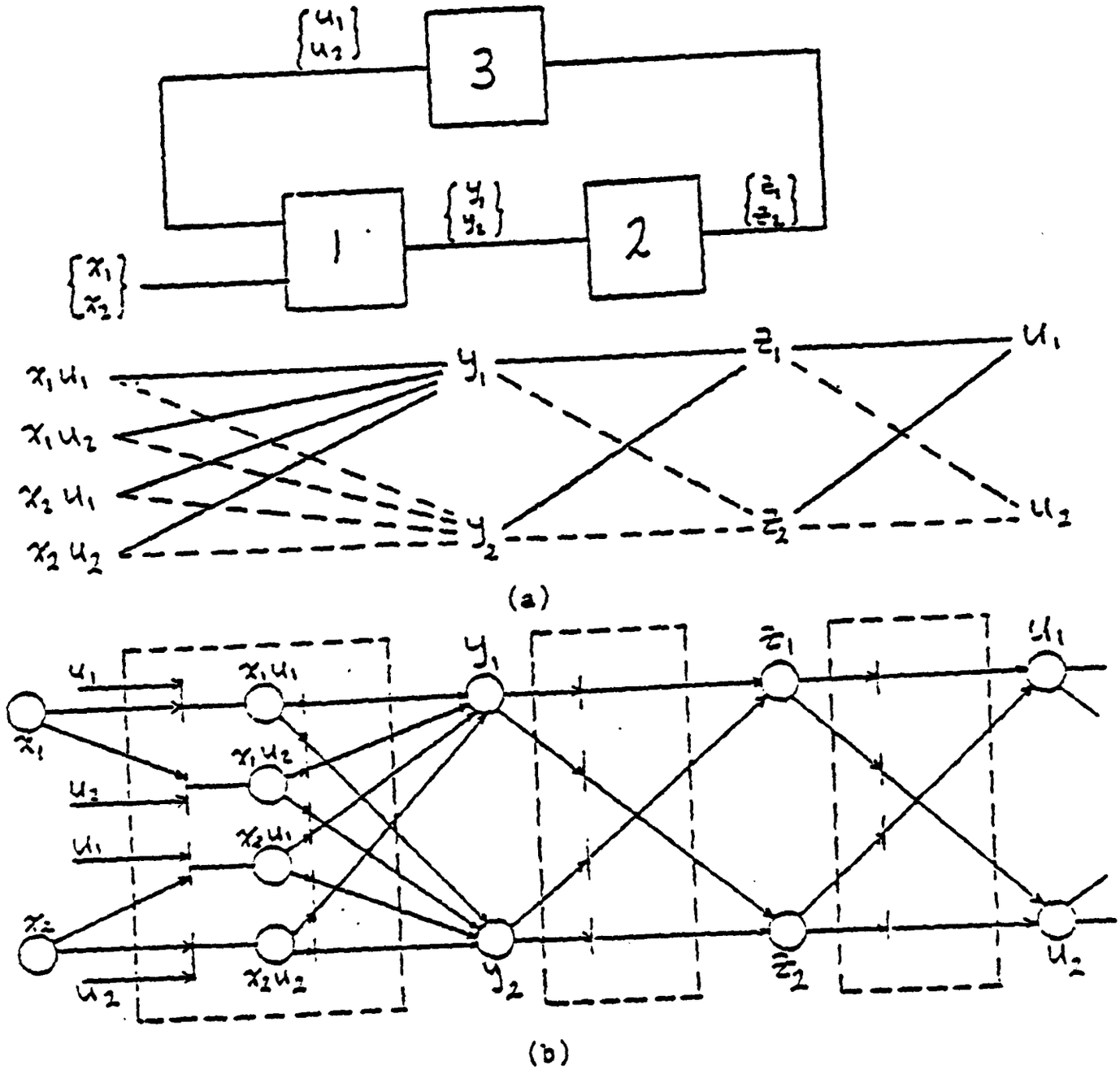


Figure 8.3

The above model can be used as a design for testability (DFT) model. In [BAL 84], some DFT models based on fault trees were discussed. The complexity of which is proportional to 2^n, where n is the number of modules. On the other hand, the complexity of GSPN is linearly proportional to n. Also, local changes in the system would require only corresponding local changes in the model.

CHAPTER 9

CONCLUSIONS

The intent of this research was to develop analytical models for parallel processing systems. Because of the nature of such systems, the model has to handel such phenomena as parallel synchronous-asynchronous operations, sequential operations, contention for multiple resources, and queuing. The models previously developed, all of which are based on product form queuing networks (PFQN), are restricted to a certain type of parallel operations or another, namely, synchronous or asynchronous operations. We considered the generalized stochatic Petri Nets (GSPNs) as an alternative modeling tool to model such systems. Yet a GSPN model rapidly becomes intractable due to the state space explosion. Therefore, a hierarchical model was developed which utilizes both GSPNs and PFQNs. A GSPN was used to model the system workload, which comprises such activities as sequential, and parallel synchronous-asynchronous operations. And a PFQN was used to model contention and queuing for the system resources.

In the second part of the dissertation, the analysis of GSPNs was considered. A general method of analysis, based on identifying an isomorphism between GSPNs and stochastically discontinuous Markov processes, was developed. This method, though was shown to be more general than previously proposed methods, still needs the generation of the reachability set of the GSPN. Which grows rapidly with the number of tokens in the initial marking and the structure of the GSPN. Therefore two

techniques for reducing analysis complexity were developed.

The first technique deals with the decomposition of the initial marking into several initial markings under which a simplified analysis of the GSPN can be obtained. For example, the examples given at the end of chapter 5 with several tokens initially in pl can be analyzed first with only one token in pl to determine the ergodic classes and transient markings under immediate transitions, which then can be obtained for the general case when there is more than one marking in pl. This technique can be applied to reduce the analysis complexity of a GSPN model of the system workload when the multiprogramming level is high.

Thesecond technique is based on aggregation and reduction at the GSPN level. Where subnetworks of immediate transitions can be aggregated or reduced. This reduces the structure complexity of GSPNs.

Approximate analysis of SPNs by multiple time scale decomposition and lumping were then considered. It was shown that the analysis of singularly perturbed SPNs, with transition firing rates of different orders of magnitude, can be reduced to that of a hierarchical sequence of aggregated subnetworks, each of which is valid at a certain time scale. The Approximate lumping of SPNs was also considered, and an example of obtaining a lumped product form SPN from a non-product form SPN was given.

Finally the application of GSPNs to model systems Reliability, Maintainability, and fault diagnosis was considered. And it was shown than such models are much more easier to construct in a compact form, especially for systems with a large

number of redundant components, than the existing Markov chain models.

BIBLIOGRAPHY

ADAM 70 Adams, D. A. "A Model for Parallel Computations," Parallel Processor Systems, Technologies and Applications ,Hobbs (ed.), Spartan Books, New York, 1970.

BAE 77 Baer, J. L., and J. Jensen "Simulation of Large Parallel Systems: Modeling of Tasks," Measuring, Modeling, and Evaluating Computer Systems, H. Beilner, and E. Gelenbe (eds.), North-Holland, 1977.

BAER 73 Baer, J. L. "A Survey of Some Theoretical Aspects of Multiprocessing," ACM Computing Surveys, vol. 5, No. 1, March 1973, pp. 31-80.

BAL 84 Balaban, H. S., and W. R. Simpson "Testability / Fault Isolation by Adaptive Strategy," Proceedings of the 1984 Reliability and Maintainablility Symposium, IEEE, 1984.

BAR 77 Bard, Y. "A Charactrization of VM/370 workloads" Modeling and Performance Evaluation ofComputerSystems, H. Beilner, and E. Gelenbe (eds.), North-Holland, 1977.

BAR 78 ----- "The VM/370 Performance Predictor," ACM Computer Surveys, vol. 10, pp 333-342, 1978.

BAR 79 ----- "Some Extensions to Multi-class Queuing Networks Analysis," Performance ofComputer Systems, M. Arato, A. Butrimenko, and E.Gelenbe (eds.), North-Holland, 1979.

BAU 78 G. M. Baudet "Asynchronous Iterative Methods for Multiprocessors," Journal of the ACM, 25(2), pp 226-244, 1978.

BUZ 71 Buzen, J. P. "Queuing Network Models of Multiprogramming" Ph.D. Dissertation, Harvard Univ., Cambridge, MA, 1971.

BUZ 80 Buzen, J. P., and P. J. Dennings " Measuring and Calculating Queue Length Distributions" Computer, vol. 13, pp. 33-44, 1980.

CER 72 Cerf, V. "Multiprocessor Semaphores, and a Graph Model of Computations," Ph. D. Thesis, UCLA, April 1972.

CHA 77 Chandy, K. M., J. H. Howard, and D. F. Towsley "Product Form and Local Balance in Queuing Networks," Journal of the ACM, vol. 24, pp 250-263, 1977.

CHA 81 Sauer, C. H., and K. M. Chandy Computer Systems Performance Modeling, Prentice-Hall, 1981.

CHI 85 Chiola, G. "A Software Package for the Analysis of Generalized Stochastic Petri Nets," International Conference on Timed Petri Nets, IEEE, July 1985

COD 83a Coderch, M., A. S. Willsky, S. S. Sastry, and D. A. Castanon "Hierarchical Aggregation of Linear Systems with Multiple Time Scales," IEEE Transactions on Automatic Control, vol. AC-28, No.11, November 1983.

COD 83b ----- "Hierarchical Aggregation of Singularly Perturbed Finite State Markov Processes," Stochastics, vol. 8, pp 259-289, 1983.

CON 63 Conway, M. E. "A Multi-Processor System Design" Proceedigs of the AFIPS 1963 FJCC, pp 136-146, 1963.

COU 77 Courtois, P. J. Decomposability: Queuing, and Computer System Applications, Academic Press, New York, 1977.

DEL 83 Delebecque, F., and J. P. Quadrat "Optimal Control of Markov Chains Admitting Strong and Weak Interactions" Automatica, vol. 17, pp 281-296, 1981.

DEL 84 Delebecque, F., J. P. Quadrat, and P. V. Kokotovic, " A Unified Approach of Aggregation and Coherency in Networks and Markov Chains," Private Communications, 1984.

DIJ68 Dijkstra, E. W. "Cooperating Sequential Processes," Programming Languages, F. Genuys (ed.), Academic Press, 1968.

DOL 83 Dolny, R. E., R. E. Fleming, and R. L. De Hoff "Fault-Tolerant Computer Systems Design Using GRAMP," Proceeding of the 1983 Reliability and Maintainability Symposium, IEEE, 1983.

DOO 42 Doob, j. L. "Topics in the theory of Markov Chains", Transactions of the Americam Mathematical Society, vol. 52, pp. 57-64, 1942.

DOO 53 ----- Stochastic Processes, Wiley, New York, 1953.

DYN 65 Dynkin, A. A. Markov Processes, Springer-Verlang, Berlin, 1965.

FLE 84a Fleming, R. E., L. J. Dolny, R. L. De Hoff "Fault-Tolerant Design-To-Specs with GRAMP and GRAM," Proceedings of the 1984 Reliability and Maintainability Symposium, IEEE, 1984.

FLE 84b Fleming, R. E., J. Josselyn, L. J. Dolny, R. De Hoff

"EarlyDesign Tradeoffs ofTest StrategyEffectiveness and Support Cost," Proceedings of AUTOTESTCON, 1984.

FLE 85 ----- "Complex System RMA&T Using Markov Models," Proceedings of the 1985 Reliability and Maitainablity Symposium, IEEE, 1985.

FLO 84 Florin, G., and S. Natkin "An Ergodicity Condition for a Class of Stochastic Petri Nets" Fifth Workshop on Application and Theory of Petri Nets, Arhus-Denmark, june 1984.

GOS 71 Gostelow, K., and V. Cerf, G. Estrin, and S. Volansky "Proper Termination of Flow of Control in Programs Involving Concurrent Processes," Proceedings of the ACM National Conference, 1972.

HEI 82 Heidelberger, P., and K.S. Trivedi "Queuing Network Models for Parallel Processing with Asynchronous Tasks," IEEE Transactions on Computers, vol. C-31, pp 1099-1109, November 1982.

HEI 83 ----- "Analytical Queuing Models for Programs with Internal Concurrency," IEEE Transactions on Computers, vol C-32, January 1983.

HEI 84 Heidelgerger, P., and S. Lavenberg "Computer Performance Evaluation Methodology," IEEE Transactions on Computers, vol. C-33, December 1984.

HER 75 Herzog, U., L. Woo, and K. M. Chandy "Solution of the Queuing Problems by a Recursive Technique," IBM Journal of Research and Development, vol. 19, May 1975.

HER 79 Herzog, U., W. Hoffmann, and W. Kleinoder "Performance Modeling and Evaluation for Hierarchically Organized Multiprocessor Computer Systems," Proceedings of the 1979 International Conference on Parallel Processing, IEEE, 1979.

KAI 84 Kai Hwang, and F. Briggs Computer Architecture and Parallel Processing, McGraw-Hill, 1984.

KAR 66 Karp, R. M., and R. E. Miller "Properties of a Model of Computations: Determinacy, Termination, Queuing," SIAM Journal of Applied Mathematics, XIV (November 1966).

KEI 78 Keilson, J. Makov Chain Models-Rarity and Exponentiality, Springer-Verlang, New York, 1978.

KEL78 Kelly, F. P. Reversibility and Stochastic Networks, Wiley, New York, 1978.

KEM 60 Kemeny, J. G., and J. L. Snell Finite Markov Chains,

Van Nostrand-Reinhold, Princeton, New Jersy, 1960.

KLI 75 Kleinrock, L. Queuing Systems, Vol. I: Queuing Theory, Wiley, New York, 1975.

KLI 76 ----- Queuing Systems, Vol. II: Computer Applications, Wiley, New York, 1976.

KUN 76 Kung, H. T. "Synchronous and Asynchronous Parallel Algorithms for Multiprocessors," Algorithms and Complexity: New Directiona and Recent Results, J. F. Traub (ed.), Academic Press, New York, 1976.

KUN 80 ----- "The Structure of Parallel Algoriths," Advances in Computers, vol. 19, M. S. Yovits (ed.), Academic Press, 1980.

MAE76 Maekawa, M., and D> L. Boyd "Two Models of Task Overlap within Jobs of Multiprocessing Multiprogramming Systems," Proceedings of the 1976 International Conference on Parallel Processing, IEEE, 1976.

MAR 84 Marsan, M. A., G. Balbo, and G. Conte. "A Class of Generalized Stochastic Petri Nets for the Performance Evaluation ofMultiprocessorSystems,"ACM Transactions on Computer Systems, vol. 2, No. 2, May 1984.

MOL 81 Molloy, M. K. "On the Integration of Delay and Throughput measures in Distributed Processing Models," Ph.D. Dissertation, UCLA, 1981.

MOL 82 ------ "Performance Analysis Using Stochastic Petri Nets," IEEE Transactions on Computers, vol. C-31, September 1982.

MUR 77 Murata, T. "State Equations, Controlability and Maximal Matching of Petri Nets," IEEE Transactions on Automatic Control, June 1977.

NAT 80 Natkin, S. Les Reseaux de Petri Stochastique, These de Docteur Ingenieur, CNAM-PARIS, June 1980.

PET 74 Peterson, J. L., and T. H. Bredt "A Comparison of Models of Parallel Computations," Proceedings of the IFIP Congress, 1974.

PET 75 Peterson, J. L., and W. Bulgren "Studies in Markov Models of Computer Systems," Proceedings of the ACM Annual Conference, 1975.

PET 81 Peterson, J. L. Petri Net Theory and the Modeling of Systems, Prentice-Hall, 1980.

PHI 81 Phillips, G. Randolph, and Petar V. Kokotovic "A Singular Perturbation Approach to Modeling and Control

of Markov Chains," IEEE Transactions on Automatic Control, Vol. AC-26, No. 5, October 1981.

PIP 84 Piptone, F. "An Expert System for Electronics Trouble Shooting Based on Function and Connectivity," Private Communication, 1984.

PRI 75 Price, T. G. "Models of Multiprogrammed Computer Systems with IO Buffering," Proceedings of the Fourth Texas Conference of Computer System, University of Texas, Austin, 1975.

RAM 74 Ramchandani, C. "Analysis of Asynchronous Concurrent Systems by Timed Petri Nets," Ph.D. Thesis, MIT, 1974, Project MAC report # MAC-TR-120.

REI 80 Reiser, M., and S. S. Lavenberg "Mean Value Analysis of Closed Multi-chain Queuing Networks," Journal of the ACM, vol. 27, pp 313-322, 1980.

TOM 84 Thomasian, A., and P. Bay "Queuing Network Models for Parallel Processing of Task Systems," Proceedings of the 1983 International Conference on Parallel Processing, IEEE, 1983.

TOW 75 Towsley, D. "Local Balance Models of Computer Systems," Ph.D. Dissertation, Department of Computer Science, University of Texas, Austin, 1975.

TOW 78 Towsley, D., K. M. Chandy, and J. C. Browne "Models for Parallel Processing within Programs: Applications to CPU:IO and IO:IO Overlap," Communications of the ACM, vol.21, pp 821-831, 1978.

DISTRIBUTION LIST

M. E. NUNN	CODE 9304	NAVAL OCEAN SYSTEMS CENTER SAN DIEGO, CA. 92152
W. KEINER	CODE F305	NAVAL SURFACE WEAPONS CENTER DAHLGREN VA. 22448-5000
J. KUNERT	CODE 92512	NAVAL AIR ENGINEERING CENTER LAKEHURST N.J.
T. COPPOLA	CODE RBET	ROME AIR DEVELOPMENT CENTER GRIFFISS AFB, ROME, N.Y.
B. ASHMORE		ATAC, INC. 1200 VILLA ST. MOUNTAINVIEW, CA. 94041
NAVAL AIR SYSTEMS COMMAND(LIBRARY) AIR-9330	14 COPIES 1 COPY	WASHINGTON, D.C. 20361
R. TENNEY		ALPHATECH, INC. 111 MIDDLESEX TURNPIKE BURLINGTON, MA. 01803